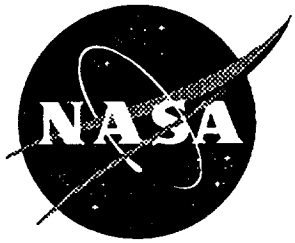
NASA Contractor Report 201632



Documentation for a Structural Optimization Procedure Developed Using the Engineering Analysis Language (EAL)

Carl J. Martin, Jr.
Lockheed Martin Engineering & Sciences Company, Hampton, Virginia

Contract NAS1-19000

October 1996

National Aeronautics and
Space Administration
Langley Research Center
Hampton, Virginia 23681-0001

Table of Contents

INTRODUCTION

This report describes a structural optimization procedure developed for use with the Engineering Analysis Language (EAL) finite element analysis system (reference 1). The procedure is written primarily in the EAL command language. Three external processors which are written in FORTRAN generate equivalent stiffnesses and evaluate stress and local buckling constraints for the sections. Several built-up structural sections were coded into the design procedures. These structural sections were selected for use in aircraft design, but are suitable for other applications. Sensitivity calculations use the semi-analytic method, and an extensive effort has been made to increase the execution speed and reduce the storage requirements. There is also an approximate sensitivity update method included which can significantly reduce computational time. The optimization is performed by an implementation of the MINOS V5.4 linear programming routine (reference 2) in a sequential linear programming procedure.

This document is divided into three sections. The first is a description of the EAL optimization procedure and the various routines involved. The second section describes the various data inputs and special processors developed for the optimization process. The final section describes the application of this process to a structural design problem. A ten-bar truss is used as an example case for the optimization procedure. The output from the linear programming routine is also discussed in this section. The appendix contains the listing of the entire EAL optimization runstream procedure, and the comments contained within the runstream provide additional documentation.

SECTION 1: DESCRIPTION OF THE OPTIMIZATION PROCEDURE

The structural optimization procedure described in this report is a combination of EAL analysis and database processors, EAL command language routines, and three special processors (LIP4, MBED, and CONS) written in FORTRAN and linked to the EAL executable code. This section provides an explanation of the overall procedure logic and a brief description of each runstream procedure. A listing of the complete optimization runstream is contained in the appendix. This structural optimization process is written in EAL command language and uses EAL utility processors extensively for the matrix operations and data manipulation. An understanding of the EAL command language and structural analysis processors is necessary for interpreting the details of this procedure. This section is intended only to provide an overview of the methods used and a brief explanation of each procedure.

1.1 Overview of the Optimization Procedure

A flow chart of the optimization procedure is shown in figure 1.1. This section provides overviews of the optimization methodology and the functions of the individual runstream procedures. A brief explanation of each major element in the flow chart follows.

Define model and optimization problem

This block represents the data which define the structural analysis and the optimization problem. This is the only portion of the optimization runstream which should require modification by the user. The finite element model definition is contained in the INIT MODL runstream procedure. The INIT MODL procedure is simply an EAL static analysis runstream through the processor TAN with the design regions defined as element groups. Various optimization control parameters, such as number of iterations and initial move limits, are defined in the SET PARA procedure. The DESV DEFI procedure contains tables COMP GRP, COMP DV, DEFL DEFI, and Exx LENG which describe the design regions, design variables, and constraints. These procedures are executed once at the start of the optimization process.

Static analysis and constraint evaluation

The first step in any optimization iteration is a static analysis and constraint evaluation with the latest values of the design variables. This is controlled by procedure CALC STAT. CALC STAT calls the procedure SECT INIT which executes the section stiffness processor MBED which generates new element stiffnesses based on the current design variable data. This data is then embedded into the appropriate element efile locations using EAL processor EI1. CALC STAT then controls the execution of a static structural analysis by calling the appropriate EAL processors. Procedure CALC N is called to compute the element stress resultants. Element stress and local buckling constraints are then calculated in the processor CONS. Displacement constraints are then added to the STRS CONS lset 1 dataset in the procedure DISP CONS.

Evaluate penalty

A penalized objective function, or simply a penalty, based on the violated constraints is calculated in procedure CALC PEN. The penalty is calculated as follows:

$$\text{penalty} = \text{objective function} + \text{KPEN} * \Sigma\,(\text{violated constraints})$$

where KPEN is a penalty parameter set by the user in procedure SET PARA. This penalty is used by the linear programming routine in an L1 Penalty formulation and also to evaluate the latest optimization cycle. For the L1 Penalty formulation to converge, the value of KPEN must be larger than the largest constraint Lagrange multiplier. However, a very large value will reduce the move limits too rapidly. If the penalty increases from one cycle to the next, the latest optimization results are discarded and the optimization proceeds from the previous step (**Reset design**). The procedure is designed to continually reduce this penalty as the optimization progresses and the constraints become better satisfied. An

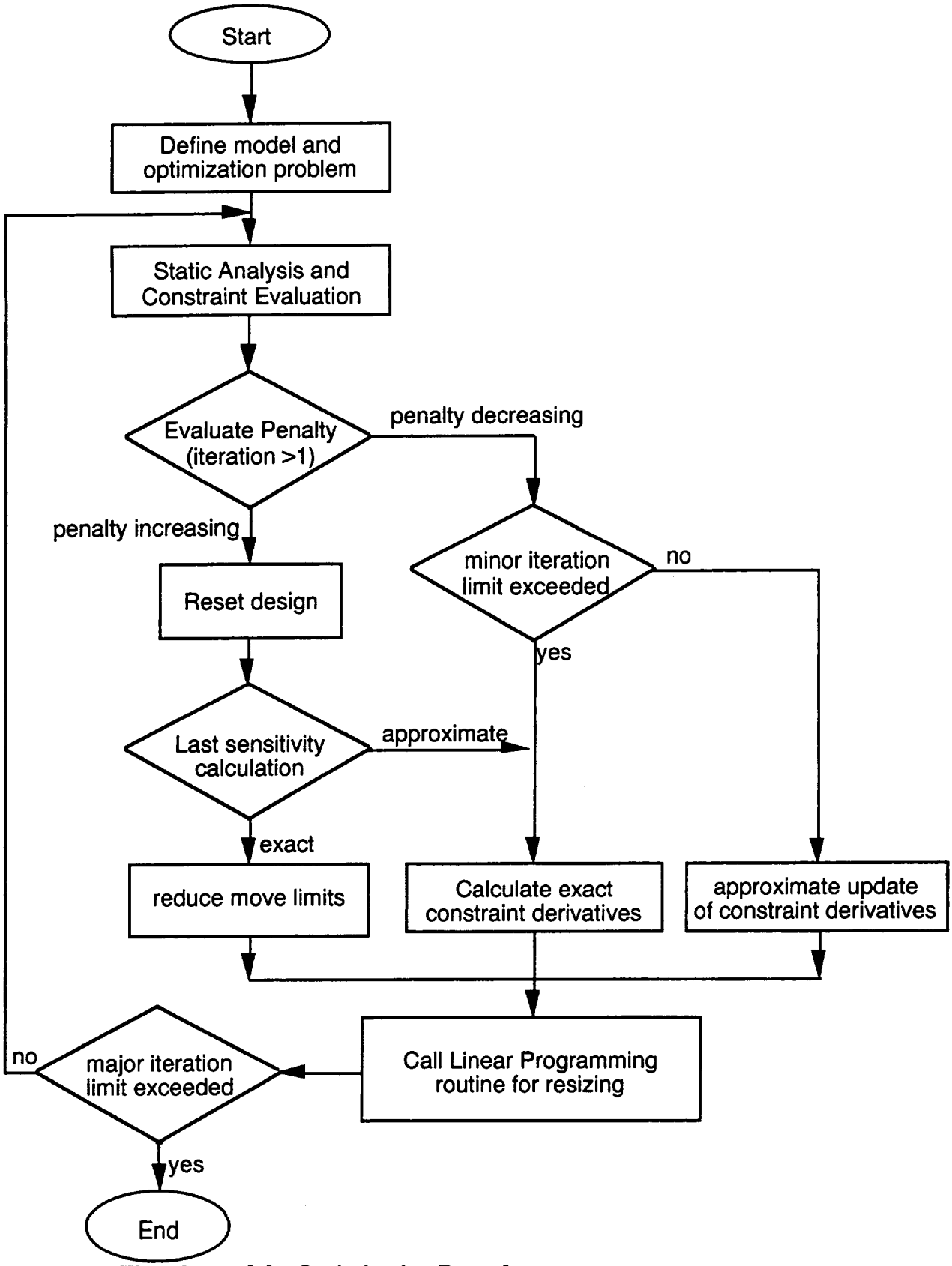


Figure 1.1 Flowchart of the Optimization Procedure

increase in the penalty after an optimization cycle generally indicates the current linear constraint sensitivities are of poor quality over the move limit range. The penalty is calculated from exact constraint calculations whereas the optimizer uses linear sensitivities to predict constraint values. A rising penalty indicates the linearized sensitivities over the move limits used in the last optimization cycle are probably not satisfactory. If these sensitivities are approximate updates (a minor iteration), they need to be recalculated with the exact sensitivity formulation (a major iteration). If the previous sensitivity calculation was exact, the move limit is cut in half (**reduce move limits**), and the linear programming processor is called again.

Calculate exact constraint derivatives.
The calculation of the objective function and constraint derivatives (a major iteration) consumes by far the most computer time. A great deal of effort has gone into making this process as efficient as possible. The sensitivity calculations are controlled by the procedure CALC SENS. The sensitivities calculated in the CALC SENS procedure are called "exact" even though there is extensive use of finite difference in their calculation. The term "exact" is used to differentiate these sensitivities from those calculated with the more approximate update method described below. The SECT INIT procedure is used to compute stiffness submatrices for both the original and perturbed values of each design variable. The derivative of the stiffness matrix is approximated by finite difference using the original and perturbed stiffness submatrices in procedure CALC DUDV. This procedure also computes the derivative of the displacement vector using the semi-analytical method. A perturbed displacement vector is approximated using a Taylor's series expansion. Stress resultants of the perturbed design are computed in the procedure CALC DELN, and constraints are computed using these stress resultants and the perturbed design variable using processor CONS. Displacement constraints are evaluated in the procedure DISP CONS. In addition, CALC MASS is called to compute the perturbed objective function. Sensitivities for both the constraints and objective function are computed by finite difference. An exact sensitivity analysis will be computed when; 1) the number of approximate iterations exceeds parameter MMINOR, 2) the penalty from the last approximate iteration increases, or 3) no sensitivities exist (the first optimization iteration).

Approximate update of constraint derivatives
The optimization procedure allows the option of using an approximate constraint sensitivity update method (a minor iteration) instead of an exact sensitivity calculation for each linear programming optimization iteration. The procedure UPDA GRAD uses the values of the constraints from the most recent static analysis and constraint sensitivity values to update the constraint gradients. The method is described in detail in reference 3. Approximate iterations will be performed as long as the minor iteration limit (parameter MMINOR) has not been exceeded and the computed penalty is decreasing. An increasing penalty indicates the quality of the approximate sensitivities is no longer acceptable.

Call Linear Programming routine for resizing
The LIP4 processor implements the MINOS V5.4 linear programming routines for the optimizer. The constraints and constraint sensitivities are assembled for use by LIP4 in the procedure ADS CONS. See the description of the LIP4 processor in section 2.5 for more information on the linear programming implementation. The outputs of the LIP4 processor are new design variable values and linear approximations of the constraint values. This new design is contained in the dataset X ADS and will be copied to the COMP DV dataset in preparation for the next iteration.

Since there is no convergence criterion implemented, the optimization procedure will continue until the number of major iterations reaches the limit. It is left to the user to evaluate the output to determine when satisfactory convergence has been achieved. The process may be restarted from any design point by using the complete design history which is stored in Library 3.

1.2 Optimization Runstream Procedures

The following is an alphabetical listing of the various procedures in the optimization runstream. It is designed to help the user understand the flow of the process. It is recommended that the user examine the comments in the procedure listing for more detailed information.

ADS CONS
Assembles constraints and constraint sensitivities into table formats to be read by the linear programming processor LIP4. The ADS in the name once referred to ADS optimization routines, but this procedure now formats data for use by LIP4.
Called by: DRIV OPT
Calls: none

ADS INIT
Initializes data tables such as X ADS (design variables) and XB ADS (design variable bounds) for use by linear programming processor LIP4. This routine determines required dataset sizes and reserves the space by creating the tables. The ADS in the name once referred to ADS optimization routines, but this procedure now formats data for use by LIP4.
Called by: DRIV OPT
Calls: none

CALC DELN
Calculates perturbed displacements based on displacement derivatives from CALC SENS using a Taylor's series expansion. It also computes stress resultants based on perturbed design variables and displacements.
Called by: CALC SENS
Calls: CALC N

CALC DUDV
Calculates displacement derivatives with respect to design variables by computing dK/dV of stiffness matrix subset using finite difference and then using the semi-analytical sensitivity formulation.
Called by: CALC SENS
Calls: none

CALC MASS
Computes the mass of the structure for use as the objective function from density, cross-section, and geometry data contained in the element efile.
Called by: CALC STAT, CALC SENS
Calls: MASS GRP1, MASS GRP2

CALC ML
Computes the design variable move limits before each call to the linear programming processor LIP4. This routine combines the upper and lower bounds from the COMP DV dataset and the move limit fraction specified by parameter ML.
Called by: DRIV OPT
Calls: none

CALC N
Calculates the stress resultant for the active design groups of the various element types using EAL processor ES.
Called by: CALC STAT, CALC DELN
Calls: STRS GRP

CALC PEN
Calculates the penalty value using the violated constraints.
Called by: DRIV OPT
Calls: none

CALC SENS
Controls the sensitivity calculations for the design variables. Loops through the design variables, perturbs each, and controls sensitivity calculations of the displacements, constraints, and mass using the semi-analytical method and finite difference.
Called by: DRIV OPT
Calls: SECT INIT, CALC MASS, CALC DUDV, DISP CONS, ELEM MBED

CALC STAT
Performs static analysis of structure using the current values of the design variables. Calculates the stress resultants and the structural mass.
Called by: DRIV OPT
Calls: SECT INIT, CALC N, CALC MASS

CLEA NL01
This routine deletes datasets which are no longer needed in library 1 (L01) and then packs the library. This helps keep disk usage to a minimum.
Called by:
Calls: none

DESV DFN
In this routine the user inputs data tables which define the optimization problem. These tables are the COMP GRP, COMP DV, DEFL DFN, Exx LENG ngrp, and LCAS SET datasets which are described in section 2.1 of this documentation. This routine is meant to be modified by the user.
Called by: DRIV OPT
Calls: none

DISP CONS
Gets the displacement limits from DEFL DEFI dataset and also the computed displacements from STAT DISP and calculates the constraint value or constraint sensitivity. These are then written to the STRS CONS or STRS DCON datasets.
Called by: DRIV OPT, CALC SENS
Calls: none

DRIV OPT
This procedure controls the optimization process. This is the top level runstream procedure which controls major and minor iterations, computes penalties, and calls the linear programming routine.
Called by: none
Calls: SET PARA, DESV DEFN, INIT MODL, ADS INIT, CALC STAT, CALC SENS, CALC ML, ADS CONS, CALC PEN, CLEA NL01, DISP CONS, UPDA GRAD

ELEM MBED
This procedure takes section and element data produced by processor MBED and uses the EAL EI1 processor to embed the section data into the element efile.
Called by: SECT INIT, CALC SENS
Calls: none

INIT MODL
This procedure contains the finite element model; nodes, elements, boundary conditions, and loads, for the optimization problem. This procedure is essentially an EAL static analysis through processor TAN and is one of three which will require user modification.

Called by: DRIV OPT
Calls: none

MASS GRP1
Calculates mass of one dimensional element types by extracting element length, cross-sectional area, and density from the efile and then sums the mass of the elements of that type.

Called by: CALC MASS
Calls: none

MASS GRP2
Calculates mass of two dimensional element types by extracting element area and unit weight from the efile and then sums the mass of the elements of that type.

Called by: CALC MASS
Calls: none

SECT INIT
This procedure controls the calculation of element stiffnesses from the design variables, the placement of these stiffnesses in the element efile, and the creation of stiffness submatrices for sensitivity calculations. The stiffnesses for elements related to the current design variable are calculated by processor MBED which outputs datasets containing the stiffness and mass data as well as a table of the affected elements. This data is written to the efile by procedure ELEM MBED. Element stiffnesses for affected elements and stiffness submatrices are then computed by EAL processors EKS and LSK.

Called by: CALC STAT, CALC SENS
Calls: ELEM MBED

SENS FILT
Selects element groups for which sensitivities are to be calculated when GLIM <0.0. If the maximum constraint value for a group is less than GLIM, the group is marked as inactive.

Called by: DRIV OPT
Calls: none

SET PARA
Initializes various optimization and sensitivity calculation parameters such as finite difference step size and number of iterations. This routine is meant to be modified by users.

Called by: DRIV OPT
Calls: none

STRS GRP
This procedure loops through all the groups of the specified element type and determines which are active design groups and commands processor ES to calculate stress resultants for these groups.

Called by: CALC N
Calls: none

UPDA GRAD
This procedure calculates updated constraint gradients using an approximate method. The formulation of the approximate gradient update is contained in reference 3. The approximate gradient update is computed during a "minor" iteration and requires much less computer time than the full sensitivity calculation or "major" iteration.

Called by: DRIV OPT
Calls: none

SECTION 2: DATA AND PROCESSORS UNIQUE TO THE OPTIMIZATION PROCEDURE

This section of the report describes the various inputs, element section types, and special processors which have been defined specially for this optimization procedure. Explanations for the remaining runstream elements are contained in the EAL users manual. This section serves as a reference guide to the datasets and parameters in the optimization procedure and, in conjunction with section 3, guides the user in implementing an optimization problem. Five major topics are covered: 1) the required input datasets, 2) the optimization control parameter settings, 3) the section definitions for the various element types, 4) the materials data file, and 5) the special processors implemented for this procedure.

2.1 Optimization Problem Definition Datasets

COMP GRP

The COMP GRP dataset defines the material and section type for a design group and also links the group to the corresponding design variables in dataset COMP DV. The table parameters for COMP GRP are NI= 4 and NJ=the number of active design groups. An active design group is any EAL element group subject to design and constraint evaluation. A group becomes an active design group by being listed in COMP GRP. The first column, as shown in the example dataset below, contains the component number. The component number associates the active design groups with the related design variables in the COMP DV table. More than one COMP GRP entry can reference the same component. In the example, E33 group 1 and E43 group 1 are in the same component. This would result in the two groups having the same design variables and section properties. The second column is the material and section type indicator. For certain types of elements there are various construction types which use different stiffness, stress, and weight formulations. Plate elements (E33 and E43) have four different section models; 1) isotropic honeycomb (0<type<1000), 2) corrugated shear web (1000< type<2000), 3) composite honeycomb (2000<type<3000), and isotropic flat plate (3000<type <4000). See section 2.3 for an explanation of the various section types. The thousands place in the column two entry specifies the section type. The remaining three digits specify the material. For example the first component (J=1) uses section 1 and material 1. The sixth component (J=8) uses section 2 and material 3. The third column is the EAL element type. The optimization procedure will only handle E21, E23, E33, and E43 type elements. Other element types may be contained in the model, and they will retain their initial material and section data throughout the optimization. The fourth column is the element group number as defined by the ELD/GROUP command. Note that E43 group 3 is not defined in COMP GRP. Elements in this group will retain their initial section properties as defined in TAB/SA section definition. An example COMP GRP dataset is shown below.

```
TABLE (NI=4,NJ=10,TYPE=0): COMP GRP
$  IN TABLE COMP GROUP NI=4, NJ=TOTAL NUMBER OF ELEMENT GROUPS
$     I=1; COMPONENT NUMBER - FOR ASSOCIATION IN COMP DV
$     I=2; MATERIAL AND SECTION TYPE INDICATOR
$     I=3; ELEMENT TYPE (E21,E43,ETC)
$     I=4; ELEMENT GROUP NUMBER
 J=1:  1       1      E33    1
 J=2:  2       2      E33    2
 J=3:  1       1      E43    1
 J=4:  2       2      E43    2
 J=5:  5       5      E21    1
 J=6:  3       5      E23    2
 J=7:  4       5      E23    1
 J=8:  6       1003   E43    4
 J=9:  7       1004   E43    5
 J=10: 8       1      E43    6
```

COMP DV

The COMP DV dataset contains the current value of the design variables, lower and upper bounds, scale factors, and a linking to the COMP GRP dataset. The sample COMP DV dataset shown below is related to the previous example COMP GRP dataset. The first column of COMP DV is the component number. The component number relates the design variables to the element groups as defined in COMP GRP. The second column is the design variable number in that component. For example, component 5 (J=7,10) has four design variables. From the COMP GRP dataset it can be determined that the component 5 is an E21 beam element. The beam in the optimization model is a symmetric wide flange beam which has four design variables; 1) flange width, 2) flange thickness, 3) web height, 4) web thickness (see section 2.3). Column 3 is the current value of the design variable. This is the only COMP DV entry which will change during the optimization procedure. Column 4 and 5 are the lower and upper bounds on the design variables. These are generally established by physical considerations such as minimum gages. The bounds during any call to the optimizer are established by a combination of these bounds and the move limits (parameter ML). Columns 6 and 7 are coefficients in a linear expression relating the design variable to the physical quantity it represents. Column 6 is the constant term (C0) and column 7 is the linear multiplier (C1) in the expression physical value=C0+C1*design variable. For component 3, an E23 rod element, the physical quantity is cross-sectional area which will be defined as area=0.+1.0*(0.2) where 0.2 is the design variable value, C0=0., and C1=1.0. The selection of the C0 and C1 can be beneficial in scaling the design variables and making the output easier to interpret. If the coefficient C1 is zero, then the design variable value no longer affects the design and no sensitivities are calculated for that design variable (it is no longer considered active). One use of this feature can be to deactivate areas of the structure whose designs are no longer changing such as areas which have reached minimum gage. This will reduce the problem size and run times because fewer sensitivities are required. Be careful when doing this though as constraints for the group are still evaluated and their values need to be monitored. Column 8 is an internal design variable status flag which will be set by the runstream. The user just needs to allocate space for this column at the start of the run. A sample COMP DV dataset is shown below.

```
TABLE(NI=8,NJ=41): COMP DV $ design variable definition
$       I=1: Component Number - correspondence with COMP GRP
$       I=2: Design variable number in component
$       I=3: Design variable value
$       I=4: Lower bound on design variable
$       I=5: Upper bound on design variable
$       I=6: Constant C0 in linear formulation of val=C0+C1*DV
$       I=7: Coefficient C1 in linear formulation of val=C0+C1*DV
$       I=8: Status flag of design variable (controlled by program)
J= 1:  1.   1.0   1.   0.1    2.5   0.    .2     0.
J= 2:  1.   2.0   .5   0.05   1.    0.    5.     0.
J= 3:  2.   1.0   1.   0.1    2.5   0.    .2     0.
J= 4:  2.   2.0   .5   0.05   1.    0.    5.     0.
J= 5:  3.   1.0   .2   0.125  5.0   0.    1.     0.
J= 6:  4.   1.0   .4   0.125  5.0   .3    0.     0.
J= 7:  5.   1.0   1.   0.1    2.5   0.    .1     0.
J= 8:  5.   2.0   1.   0.25   3.    0.    1.     0.
J= 9:  5.   3.0   1.   0.1    2.5   0.    .1     0.
J= 10: 5.   4.0   1.   0.25   5.    0.    1.     0.
J= 11: 6.   1.0   .8   0.5    6.0   0.    .01    0.
J= 12: 7.   1.0   .8   0.5    6.0   0.    .01    0.
J= 13: 8.   1.0   1.   0.1    2.5   0.    .2     0.
J= 14: 8.   2.0   .5   0.05   1.    0.    5.     0.
```

DEFL DEFI

The deflection definition dataset, DEFL DEFI, defines deflection constraints for the optimization. The parameters for the table are NI=3 and NJ=the number of deflection constraints. Deflection constraints defined in the DEFL DEFI table are applied to all load cases and all load sets. The first column (I=1) is the node number at which the constraint is applied. The second column (I=2) is the constrained direction (a number 1 thru 6) in the nodal coordinate system, and the third column (I=3) is the deflection limit. Decimal points are required on all entries to prevent an EAL error. The deflection constraints in the optimization are posed as:

$$\text{constraint value} = \frac{\text{computed deflection}}{\text{deflection limit}} - 1.$$

which means that the second constraint in the example below would require the deflection at node 103 in the 3 direction to be greater than -100. A DEFL DEFI dataset must exist or a run time error will occur. This requires that at least one deflection constraint be defined for all problems. A dummy constraint may be required in some cases. This can be written either for a constrained node or by supplying a very large deflection limit to any node. An example DEFL DEFI dataset is shown below.

```
TABLE(NI=3,NJ=2): DEFL DEFI      $ deflection constraints
  J=1:      29.   2.    36.      $ joint 29, direction 2, 36 units
  J=2:      103.  3.    -100.    $ joint 103, direction 3, -100 units
```

Exx LENG ngrp

The Exx LENG ngrp datasets define characteristic lengths used in local buckling checks on the elements in group "ngrp" for element type "Exx". These tables are used by the processor CONS. The current applicable element types are E21, E33, and E43 - no buckling checks are performed on E23 elements. The Exx LENG ngrp datasets have table parameters NI=1 and NJ= the number of elements in the group. The table entries will then be the characteristic length to be used for constraint calculations for each element. These datasets are not required, and if they are not present CONS will use the characteristic length given by reset BLEN. Example entries are shown below.

```
TABLE(NI=1,NJ=25): E43 LENG 3
  I=1: J=1,15: 18.      $ elements 1 thru 15 have 18 in. length
       J=16,25: 24.     $ elements 16 thru 25 have 24 in. length
TABLE(NI=1,NJ=2): E21 LENG 1
       J=1: 12.
       J=2: 18.
```

LSET CASE

The LSET CASE dataset defines the number of load cases in each load set. This table has parameters NI=2 and NJ=the number of load sets. Note that only the first column (I=1) of the dataset is required - the second column is defined internally. The entries are the number of load cases for each set. See section 3.1 for a discussion of the organization of load sets and cases. An example LSET CASE dataset is shown below.

```
TABLE(NI=2,NJ=3,TYPE=0): LSET CASE
  I=1:  J=1: 3      $ load set 1 has 3 load cases
        J=2: 1      $ load set 2 has 1 load case
        J=3: 5      $ load set 3 has 5 load cases
```

2.2 Optimization Control Parameters

There are a number of parameters which control various facets of the optimization procedures. These are contained in the SET PARA procedure of the optimization control runstream. A brief explanation of each parameter is given below.

NDV — The number of independent design variables for the optimization problem. This will also be the number of rows (NJ) in the COMP DV dataset. (integer)

MMAJOR — Maximum number of major iterations to be performed during an optimization run. A major iteration includes a full recalculation of sensitivities. (integer)

MMINOR — Maximum number of minor iterations to be performed in any major iteration. A minor iteration uses an approximate method to update sensitivities. The execution of minor iterations will cease when constraint error levels cause the penalty to increase. (integer)

ML — The design variable move limits coefficient. The current value of each design variable is multiplied by parameter ML to obtain its move limits for that cycle. This value may be reduced automatically as the optimization proceeds based upon changes in the penalty value. (real)

DR — Finite difference step size fraction for the sensitivity calculations. Actual step sizes for each design variable are formed by multiplying DR and the current design variable value. (real)

RHOKS — Constraint lumping parameter. The value is used for reset RHOK when calling the CONS processor. A negative value turns off constraint lumping. A positive value utilizes K-S function constraint lumping with the value RHOKS used as the function parameter rho. (real)

NLST — Number of load sets applied to model. Each load set should have either different boundary conditions or a different temperature set. See section 3.1 for further explanation of load case and set organization. (integer)

NUMD — Number of displacement constraints applied. Should be the same as NJ in the DEFL DEFI dataset. At least one displacement constraint is required. (integer)

KPEN — The parameter used for computing the penalty for violated constraints. It is suggested that it be on the order of the objective function to start with, and it should never be less than the largest Lagrange multiplier printed by the LIP4 processor. Too small a value of KPEN will leave constraints violated and too large a value will reduce move limits too quickly. (real)

BLEN — Default characteristic buckling length used by processor CONS if an Exx LENG ngrp table is not provided. See the description of processor CONS in section 2.5 for a further description. (real)

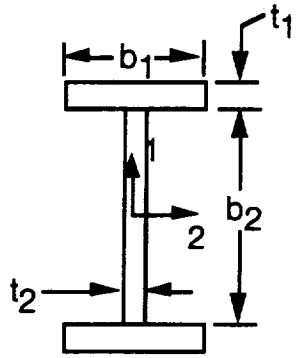
GLIM — Constraint sensititivity calculation filter limit. For GLIM < 0.0, element groups with a maximum constraint value less than GLIM will not have constraint sensitivities calculated. This can save computer time for problems with large numbers of inactive constraints. If GLIM ≥ 0.0, sensitivities are calculated for all design groups. (real)

In addition, a number of storage libraries for the data generated during the optimization procedure are defined in the SET PARA procedure. These should not require any modification.

2.3 Optimization Section Types and Design Variables

Aircraft construction in general utilizes built-up structure, and some of these construction types have been coded into the optimization procedure. The design variables are assigned to key parameters of the built-up structures, and constraints are calculated using typical design rules for the section type. This allows for reasonable modeling of entire aircraft or large components with a reasonable number of design variables and with meaningful constraints. The MBED and CONS processors generate stiffnesses and evaluate constraints for the following section types.

Wide Flange Beam (E21)



Design Variables
1) flange length b_1
2) flange thickness t_1
3) web height b_2
4) web thickness t_2

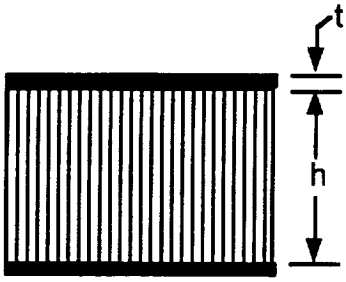
The wide flange beam element is currently the only option available for E21 elements. It is similar to the WFL section described in section 3.1.9 of the EAL Reference Manual, except that it is symmetric. There are four design variables; flange length, flange thickness, web height, and web thickness. The constraints evaluated for this section are: 1) maximum stress, 2) beam buckling, 3) flange local buckling, and 4) web local buckling. The beam orientation and reference frame are the same as for the EAL WFL section.

Axial Rod Element (E23)

Design Variable
1) cross-sectional area

The axial rod element has only one design variable - the cross-sectional area. The only constraint is the maximum stress.

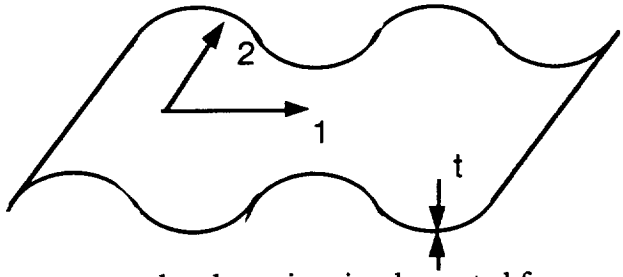
Isotropic Honeycomb Sandwich Plate (E33 and E43)



Design Variables
1) face sheet thickness t
2) honeycomb core height h

The honeycomb sandwich section is used for plate elements with section identification numbers less than 1000. The section has two design variables; face sheet thickness and honeycomb core height. Constraints utilized for this section are von Mises stress in the face sheets and local plate buckling. The buckling constraint is based on a biaxial loading with a shear interaction. The plate is considered as square with side length specified in the "Exx LENG ngrp" dataset for buckling calculations.

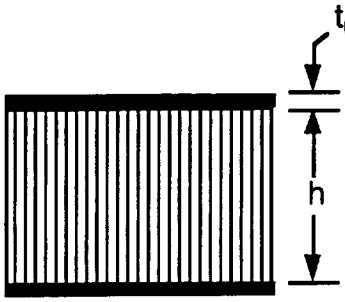
Corrugated Shear Web (E33 and E43)



Design Variable
1) thickness t

The corrugated web section, implemented for use as a shear web, is used for elements with section numbers between 1000 and 2000. The corrugations modeled are 60° circular arc corrugations with a 1.5 inch radius. The only design variable is thickness. The material axes are as defined in the figure above, and the user should take care to orient the element stress reference frame appropriately. Constraints for this section are von Mises stress, plate shear buckling, and local shear buckling. For the shear buckling constraint evaluations, a semi-infinite plate with side length specified in the "Exx LENG ngrp" dataset is assumed. Constraint equations used are derived from a study using finite element analysis by NASA Langley's Thermal Structures Branch.

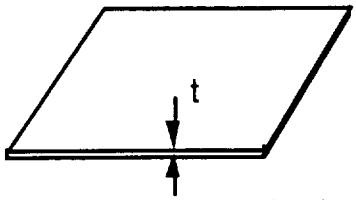
Composite Honeycomb Sandwich Plate (E33 and E43)



Design Variables
1) 0° lamina thickness (one face)
2) 90° lamina thickness (one face)
3) 45° lamina thickness (one face)
4) honeycomb core height

A composite honeycomb sandwich section is used for elements with section numbers between 2000 and 3000. The section is assumed to be balanced and symmetric, and the core height is great enough so that the location of the individual plies in the face sheets does not affect the overall bending stiffness. These assumptions along with limitations on ply angles reduces the number of design variables to four. They are: 1) total thickness of 0° plies in one face, 2) total thickness of 90° plies in one face, 3) total thickness of +45° plies in one face, and 4) the honeycomb core height. In the calculations, a -45° ply thickness equal to the thickness of +45° plies is used. Constraints used are maximum allowable strain, which can be different in tension and compression, and local panel buckling. The buckling constraint is based on a biaxial loading of an orthotropic plate with a shear interaction. The plate is considered as square with side length specified in the "Exx LENG ngrp" dataset for the buckling calculation.

Isotropic Flat Plate (E33 and E43)



Design Variable
1) thickness t

An isotropic plate section is used for elements with section numbers between 3000 and 4000. The single design variable for this section is the plate thickness, t. Constraints utilized for this section are von Mises stress and local plate buckling. The buckling constraint is based on a biaxial loading with a shear interaction. The plate is considered as square with side length specified in the "Exx LENG ngrp" dataset for the buckling calculation.

2.4 Materials File - matprop.prn

Material properties for elements being designed are contained in an external file which must be named matprop.prn and reside in the current directory. The file includes the material modulus, density, and strength data and is read by processors CONS and MBED. The file structure allows for the definition of multiple materials and temperature dependent properties. The format used is based on one used by another design tool, and thus there are many values which are not used for the EAL optimization. Example material file inputs for isotropic and composite materials are shown below. The material number corresponds to the material number specified by the material and section type indicator in the COMP GRP dataset (see section 2.1). If temperature dependent properties are used, data blocks for a material should be contiguous and in ascending temperature order. If EAL temperature datasets (NODA TEMP or TEMP Exx) are present, temperature dependent properties are expected.

The optimization procedure was initially implemented to perform preliminary design optimization for aircraft, and therefore uses some aircraft design methodology. Generally for aircraft design, stresses in isotropic materials must be below the ultimate strength at the ultimate load level and below the yield stress at limit load. The ultimate load factor is generally 1.5. The constraint processor CONS expects the input loads to be ultimate loads. Based upon these assumptions, the limiting strength value will be the lesser of the ultimate strength or the yield stress multiplied by 1.5 (the assumed load factor). While the user may not use this design philosophy, it must be understood so that appropriate material limits can be supplied in the matprop.prn file. For composite laminates, the strain limits supplied are applied to the ultimate (or input) loads. The units for the material properties supplied in this file are of the user's choice, but they must be consistent with the units in the finite element model and the loads. For honeycomb sandwich sections the user is asked to supply two densities. The density of the face sheet material (density) and the honeycomb core effective density (core density), and these two entries should have the same units. If the core density value is omitted, a core density of 2% of the face sheet density will be used.

For an isotropic material the input is as follows:

Material Number	Description of the material (optional)			
Temperature	Elastic Modulus	0.	Shear Modulus	Poisson's ratio
0.	density	ultimate strength	0.	0.
0.	yield strength	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
CTE (α)	0.	0.	0.	core density

For a composite lamina the material input is shown below. The sign used for the compressive strain limit in the materials file is not important. A limit of 0.005 and -0.005 will produce the same result.

Material Number	Description of the lamina (optional)			
Temperature	E_{11}	E_{22}	G_{12}	ν_{12}
ν_{21}	density	(ε_{11})max. tension	(ε_{11})max. comp.	(ε_{22})max. tension
(ε_{22})max. comp	0.	0.	0.	0.
(γ_{12})max.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
0.	0.	0.	0.	0.
α_{11}	α_{22}	0.	0.	core density

A sample materials file for an isotropic material is listed in section 3.1.

2.5 EAL Optimization Routine Special Processors

LIP4

This processor implements a linear programming routine or optimizer. The routine implemented at this time is MINOS V5.4. The processor reads the tables of the design variables and bounds, the objective function, the constraints, and all the sensitivities and converts this data into a form used by the linear programming subroutine. Most of the datasets are similar to those described for the EAL processor ADS. Additional optimization controls are provided through processor resets. The linear programming routine is called by processor LIP4 which returns the new design data. Constraints that cannot be satisfied within the LP approximate problem can utilize an L_inf penalty to reduce the maximum constraint violation, or an L1 penalty formulation as described in reference 5. The new design description is written to appropriate tables on exit from the processor.

Input Datasets:

X ADS i 1, i=1,2,3,....	Design variables before optimization
XB ADS i 1, i=1,2,3,....	Lower and upper bounds for X
F ADS 1 1	Objective function value before optimization
DF ADS i 1, i=1,2,3,....	Objective function gradients
G ADS 1 j, j=1,2,3,....	Constraint values before optimization (satisfied if <=0)
GB LIP 1 j, j=1,2,3....	OPTIONAL, allows constraints to be specified with lower and upper bounds on G instead of the standard <=0 form.
DG ADS i j or DG LIP i j	Constraint gradients. Dimensions will be NI = # of design variables in D.V. set i, NJ = number of constraints in constraint set j. Using the DG ADS version, the dataset can be filtered by the CUTOFF reset before optimization. Using the DG LIP version, the entire dataset is utilized by the optimizer with any CUTOFF ignored.
XNORM LIP i 1, i=1,2,3....	OPTIONAL, optionally used if CUTOFF > 0., otherwise not used. Utilized to normalize constraint gradients with respect to the design variables when determining which elements are ignored (i.e., assumed zero).

Output Datasets:

X ADS i 1, i=1,2,3,....	Design variables after optimization
F ADS 1 1	Linearized objective function value after optimization
G ADS 1 j, j=1,2,3,....	Linearized constraint values after optimization
COG LIP 1 j	Lagrange multipliers (Costate of G)
OUT LIP 1 1	Contains the value of the objective function penalty used.

Resets: (only first 4 characters significant)

NUIN	Library number for input datasets (default=1)
IPRI	Print flag (default=3)
CUTOFF	Constraint gradients (possibly normalized by XNORM) smaller than CUTOFF*max gradient are ignored (i.e., assumed zero). This can reduce the size of large problems. (default=0.)
NMDG	Maximum number of non zero terms allowed in all DG ADS i j (default=100000)
LPENALTY	Flag for penalty on the optimization. No penalty used if LPEN=0, L1 penalty used if LPEN=1, otherwise L_inf penalty used. (default=0)
KPENALTY	Objective function coefficient for penalties (i.e., KPENALTY* constraint violation is added to objective function)
ITER	Iterations (internal MINOS parameter) allowed. Ignored unless > 0 (default=0)
IEQUAL	For LPEN=1, constraints numbered IEQUAL and greater will not be included in determining the objective function penalty. (default=0).
IZERO	If IZERO is non zero, allows zero rows in DG ADS. By default error exit returned if a row is zero. (default=0)
ISCALE	Scale flag (internal MINOS parameter) (default=2)

The input dataset structure of the LIP4 processor has been designed to provide for flexibility of input that most users will never require. The optimization runstream will construct all required tables for a basic optimization problem. The structural optimization problem data as described in this documentation will be contained in X ADS 1 1, XB ADS 1 1, F ADS 1 1, DF ADS 1 1, G ADS 1 1, and DG ADS 1 1. The variable i's and j's in the dataset names allow for the user to define additional design variable groups and/or constraint groups in a simple manner. For example, an additional group of design variables could be added by creating datasets X ADS 2 1 (the additional design variable values), XB ADS 2 1 (the bounds on X ADS 2 1), DF ADS 2 1 (the sensitivities of F ADS 1 1 with respect to the design variables in X ADS 2 1), and DG ADS 2 1 (the sensitivities of the constraints in G ADS 1 1 with respect to the design variables in X ADS 2 1). If an additional group of constraints is desired, they could be placed in dataset G ADS 1 2. The sensitivities of the new constraints should then be in DG ADS 1 2 (with respect to the design variables in X ADS 1 1) and DG ADS 2 2 (with respect to the design variables in X ADS 2 1). A practical example of when a user might use this feature would be if additional, user defined constraints, such as flutter, were required. If five additional constraints were added, they would be placed in G ADS 1 2 whose parameters are NI=1 and NJ=5. The additional constraint sensitivities would be placed in DG ADS 1 2 with parameters NI=number of design variables and NJ=5 (number of constraints).

MBED

The MBED processor writes section information for the elements based on the current design variable data and section definition. The MBED processor creates a table (EMBE EFIL) containing section stiffness properties for each element in the specified design component. The design component is selected by the processor reset COMP. The reset COMP corresponds to the component number (the I=1 entry) in the COMP GRP dataset. The first six values in the EMBE EFIL dataset are the material property (segment 2) data and the remaining data are the section property (segment 4) information. This is described in section 4.5 of the EAL manual. Also created is a table of element groups (EMBE GRP) for which properties have been generated. The EMBE GRP table is used in conjunction with the EI1/EMBED command and directs where the efile data is to be embedded. In addition, a table of elements to be modified (ETAB LIST) is created for use with the SELECTION command of the EKS processor. These are the elements, specified by absolute index number, for which EKS will generate new element stiffnesses.

Stiffnesses are calculated in MBED based upon the design variable values contained in the COMP DV dataset and the section and material definitions contained in the COMP GRP dataset. Column 2 (I=2) contains the material identification number and section type (if applicable) for the component. See the description of the COMP GRP and COMP DV datasets for more details. The material identification number corresponds to material properties data contained in the external file named matprop.prn. This file contains modulus, density, and stress (or strain) limits for all the materials used in the model and must be present. The processor MBED takes the design variable data and the section and material information and computes the areas, moments of inertia, and stiffnesses required in the element efile. At present, the MBED processor does not incorporate temperature dependent material properties. The addition of this capability is planned in the near future.

Input Datasets:

COMP GRP	Design component specification table
COMP DV	Design variable definition table
DEF Exx	EAL element definition file for element type Exx
GD Exx	EAL group definition file for element type Exx
NODA TEMP iset	Nodal temperatures (optional - not currently implemented)
TEMP Exx iset	Element temperatures (optional - not currently implemented)

Output Datasets

EMBE EFIL	Section stiffness data to be embedded in efile for each element
EMBE GRP	List of element groups to which EMBE EFIL data is to be embedded
ETAB LIST	Table of elements to be processed by EKS via the SELECTION command

Resets

NUIN	Library number for input datasets (default=1)
COMP	Component number to generate tables for (no default)

Also required

matprop.prn	external file containing material properties.

CONS

The CONS processor computes the element constraints for the current design variable values using the element stress resultants (internal loads). The processor reads the datasets produced by processor ES to determine the stress resultants in each element. In addition, section and design variable data for each component are read from datasets COMP GRP and COMP DV. This information is used to evaluate element stress and stability constraints. The constraint formulations vary for each element and section type. The section definitions are described in detail in section 2.3 of this document. Two constraints are computed for each element. The first is a stress constraint. This constraint uses the maximum allowable stresses (or strains) defined in the matprop.prn material file. The stability constraint is the maximum of a number of local buckling checks performed on the section. For these checks an additional piece of information is required - a characteristic buckling length for the element. This length will be taken from the Exx LENG ngrp dataset if one exists. This dataset, which is supplied by the user and described in section 2.1, has a characteristic length entry for each element in the particular group. If this dataset is not present, the length is specified by the reset BLEN (see section 2.2). It is permissible to have certain element groups prescribed by Exx LENG ngrp datasets and others by the BLEN reset.

The CONS processor will also use EAL temperature datasets for use with temperature dependent material property evaluation. The processor uses the standard EAL temperature definition datasets as described in section 6.2 of the EAL manual. There are a few differences in usage, however. The first important difference is that if both NODA TEMP and TEMP Exx datasets are present, only temperatures from the TEMP Exx dataset will be used and the NODA TEMP data will be ignored. The second difference involves the treatment of temperature cases within a loadset. The processor will use data from the temperature case specified by reset TCASE for property evaluation for all load cases in that load and temperature set. This stems from the philosophy of the optimization runstream design. For each load set and its associated temperature set (if present), a new stiffness matrix and sensitivities are calculated. All load cases within a set will use the same stiffness matrix. A different temperature loading would requires a new stiffness matrix (temperature dependent properties) and should thus be a separate load and temperature set.

The CONS processor has two groups of output datasets. The first is a series of two column datasets Exx CON lset ngrp which contain the element constraint values for each element in group "ngrp" for load set "lset". The first column (I=1) will be the stress constraint for the Jth element in the group. The second column (J=2) is the local buckling constraint. There will be an Exx CONS dataset for each active design group. The second table created is the STRS CONS lset 1 dataset. It contains the constraint data for all the elements in a single column format. In addition, space at the end of the dataset is reserved for any deflection constraints. The data contained on the STRS CONS dataset is dependent on the reset RHOK. If RHOK is less than zero, then STRS CONS is simply a concatenation of all the Exx CONS datasets. If RHOK is positive, then the stress and buckling constraints for each group are each combined into a single continuous constraint using a K-S function. For RHOK>0, the value of RHOK is then used as the K-S function parameter rho.

Input Datasets (in library NUIN):

COMP GRP	Design component specification table
COMP DV	Design variable definition table
DEF Exx	EAL element definition file for element type Exx
GD Exx	EAL group definition file for element type Exx
NODA TEMP iset	Table of nodal temperatures (optional). See EAL manual section 6.2
TEMP Exx iset	Element temperatures (optional). See EAL manual section 6.2

Exx LENG ngrp	Table of element characteristic lengths for local buckling checks (optional)
LSET CASE	Table of number of load cases in each load set.

Input Datasets (in library NOUT):

ES Exx lset ngrp or SR Exx lset ngrp	Tables of element internal forces (stress resultants) for the elements being sized in the optimization.

Output Datasets (in library NOUT)

Exx CON lset ngrp	Table of constraint values (2 columns) for each element in group "ngrp". A table is created for each active group
STRS CONS lset 1	Table of all the stress and local buckling constraints for each element (RHOK < 0) or the K-S function constraint values for each group (RHOK > 0)

Resets

NUIN	Library number for input datasets (default=1)
NOUT	Library number for stress and constraint datasets (default=1)
SET	Load set number (default=1)
LLIB	Library number where Exx LENG ngrp datasets reside (default=1)
BLEN	Characteristic length used for local buckling calculations if Exx LENG ngrp dataset is not present. (default=24.)
RHOK	K-S function parameter (RHOK < 0, K-S function constraint lumping is not used) (default=-1)
TSET	Temperature set number for property evaluation (default=1)
TCAS	Temperature case number for property evaluation (default=1)
NOJUNK	Controls printout of warning messages when stress filtering is used (GLIM<0.0). NOJUNK=YES inhibits warning messages that stress and constraint datasets are missing for filtered element groups. (default=NO)

Also required

matprop.prn	external file containing material properties.

SECTION 3: CREATING THE OPTIMIZATION MODEL AND INTERPRETING THE RESULTS

This section of the report describes the steps involved in implementing the optimization procedure described in sections 1 and 2 for a structural design problem. The changes required to convert an EAL static analysis model into an optimization model along with the required data tables are detailed. An example problem, the commonly used ten-bar truss (reference 7), is used to illustrate the required inputs. The ten-bar truss example and the optimum element sizes are shown on the following page. A listing of the optimization output for several design cycles along with clarifying annotations is also provided. Finally, the restart capability of the optimization procedure is explained and strategies for setting and modifying the control parameters are discussed.

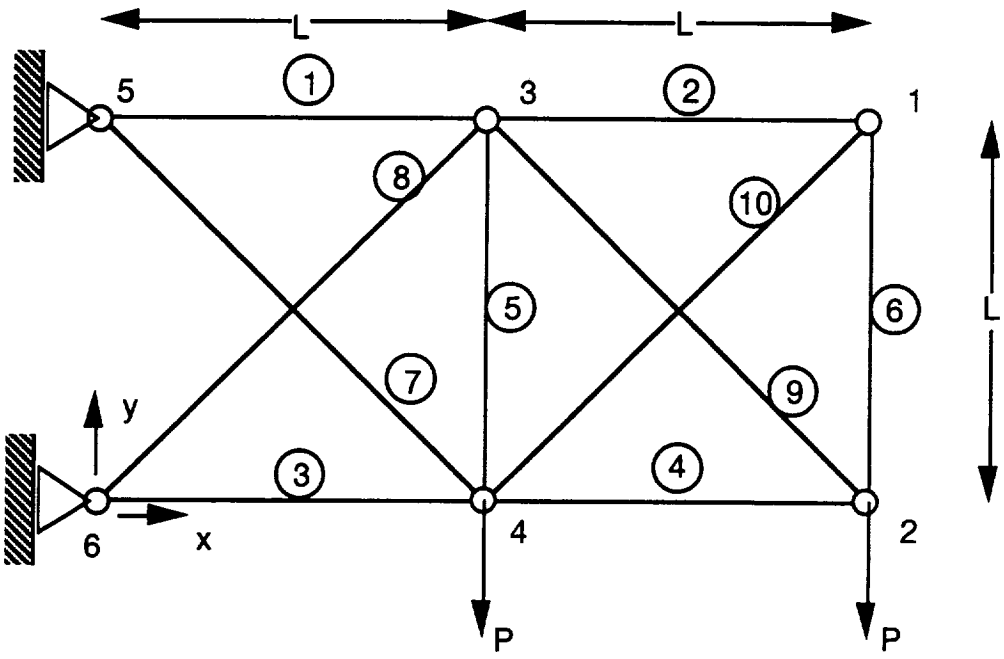
3.1 Developing an Optimization Model: The Ten-Bar Truss Example

Finite Element Model Creation and Modification

The first step in assembling the optimization problem is to create an EAL static analysis input for the structure. Details for doing this are contained in the EAL reference manual and will not be described here. It is important to obtain a satisfactory static analysis model prior to attempting the optimization as the results will only be as good as the finite element analysis. At present, not all of the EAL structural element types are implemented in the optimization procedure. Currently, only E21, E23, E33, and E43 elements may be designed. It is perfectly acceptable to include any other element type in the finite element model, but they will not be resized. There are certain section types assumed for the various element types. The E21 beam element, for example, when used as a design group is defined to have a symmetric, wide flange cross-section. The various sections are described in section 2.3, and the user must decide which is best for a given problem.

The main difference in model definition between analysis and optimization is the way element groups are assigned. In a static analysis, elements are assigned to groups by the user for convenience. For an optimization model, element groups define design regions. All elements in an active group will be associated to the same design variables and will have the same properties. Note, the previous statement only applies to groups of elements which are being designed. Elements not associated to design variables will maintain the section properties assigned to them in the EAL model definition. So each active group represents a design region, and it is possible to have different groups represented by the same design variables. This will be discussed further in the table definitions, but the important point here is that the element groups are used to define design regions. The only other potential modification to the static analysis deck will be when plate elements (E33 or E43) are used. In order to maintain maximum flexibility, the optimization routines write directly to the element efile and assume that space is reserved for sections with either coupled or laminate formats (see page 3.1-22 of the EAL reference manual). This requires that plate elements being designed reference materials which are created with SA coupled or laminate input formats so that the proper space is reserved in the efile. The section properties input in the model definition will be overwritten prior to any analysis. The actual data is not important, but using an improper format will result in an error.

The static analysis problem is contained in the INIT MODL procedure of the optimization runstream. It should contain all the static analysis data from processors TAB and ELD. In addition, the loading definitions should also be included. In a normal EAL analysis, the separation of loadings into sets and cases is generally left to the convenience of the user. The workings of the optimization procedure, however, make assumptions about the organization of load cases and sets to achieve efficient operation. The processors assume that the load set and constraint set numbers are the same; i.e., load set 3 will use constraint set 3. For this reason, all loadings which use the same constraint set should be supplied as multiple load cases in one load set. The fewest possible constraint sets should be used because new stiffness matrix sensitivities are required for each constraint set, and this is by far the most time consuming portion of the optimization. In addition to the model definition, the processors E, TAN, and SEQ(optional) should also be run in the initial model.



L = 360 in.

P = 100,000 lb

Problem Data

$E=10^7$ psi	Allowable Stress
Density = 0.1 lb/in^3	members 1-8,10: ±25,000 psi
Minimum Area = 0.1 in^2	member 9: ±75,000 psi

Optimum Mass = 1497.6 lb

Optimum Bar Areas (in^2)

Bar	Area	Bar	Area
1	7.90	6	0.10
2	0.10	7	5.80
3	8.10	8	5.51
4	3.90	9	3.68
5	0.10	10	0.14

Figure 3.1 Ten-bar Truss Example problem

The INIT MODL procedure used for the ten-bar truss example is listed below. Since each element is being designed separately, each is in its own group. The cross-section listed in the BC entry and the material data (MATC) will not be used in the optimization, but must be present to prevent errors. Note that is not necessary to give bar 9 a different material property. The materials will be assigned in the COMP GRP dataset.

Finite Element Model Definition Procedure (INIT MODL)

```
*(INIT MODL)        ENDINM
*ECHO 4
$ ENTERING: (INIT MODL)
$ INITIALIZE EAL MODEL
*NOECHO 4
*XQT TAB
   START 6 3 4 5 6              $ 6 nodes, only x&y active
   MATC: 1 1.+7 .3 .1           $ aluminum properties
   BC: 1 5.                     $ bar section properties
   JLOC                         $ list of joint locations
   1 720. 360.
   2 720. 0.
   3 360. 360.
   4 360. 0.
   5 0.   360.
   6 0.   0.
  CON=1                         $ displacement constraints
   ZERO 1 2: 5,6
*XQT ELD
 E23                            $ bar elements - each in own group
  GROUP 1:5 3
  GROUP 2:3 1
  GROUP 3:6 4
  GROUP 4:4 2
  GROUP 5:3 4
  GROUP 6:1 2
  GROUP 7:5 4
  GROUP 8:6 3
  GROUP 9:3 2
  GROUP 10:4 1
*XQT AUS
  SYSVEC: APPL FORC : I=2       $ applied loads - y forces on jnt 2&4
    J=2: -100000.
    J=4: -100000.
*XQT E
*XQT TAN
*XQT SEQ   $ optional
$
*ECHO 4
$ EXITING: (INIT MODL) $ INITIALIZE EAL MODEL
*NOECHO 4
$
*RETURN
```

Optimization Control Parameter Definitions

A number of parameters which control various aspects of the optimization process are defined in the procedure SET PARA. These parameters control such items as the number of iterations to be performed and the finite difference step size and are described in sections 2.2 and 3.3. One other item of importance is the assignment of a file name for optimizer output. EAL has a general output file which is assigned at execution. This file can be very long and repetitive for a run with many design variables and iterations. To avoid having to wade through this output to view the optimization results, optimization cycle data is piped to a separate file. This file is assigned in the SET PARA procedure with the TF OPEN command.

The SET PARA procedure for the ten-bar truss example is listed below. The number of design variables (NDV) is ten, one for each bar. Also, the number of displacement constraints is one even though there were none defined in the problem. At least one must be defined as a place holder, and a dummy displacement constraint has been created for this problem (see the next section). The number of iterations to be performed is left to the discretion of the user. There is no convergence evaluation, and the run will continue until the major iteration limit is reached. The value of RHOKS is set to -1. which specifies there will not be constraint lumping. Since there is only one element in each design group constraint lumping does not reduce the problem size. The penalty parameter KPEN is set arbitrarily at 6000. Examination of the Lagrange multipliers in the output (section 3.2) indicates a lower value would have been acceptable, but this value does achieve the optimum solution. The initial move limit fraction was set fairly high at 0.40 to hopefully speed convergence. This value is reduced as the solution progresses and the solution will generally proceed faster if a fairly large initial value is used.

Optimization Control Parameter Definition Procedure (SET PARA)

```
*(SET PARA)          ENDPAR
*ECHO 4
$ ENTERING: (SET PARA)
$ SET CONTROL PARAMETERS
*NOECHO 4
$
!NDV= 10           $ NO.OF INDEPENDENT DESIGN VARIABLES
!DR=.1E-7          $ FACTOR FOR INCREMENTING DESIGN VARIABLE
!ML=.40            $ INITIAL MOVE LIMIT ON DESIGN VARIABLES
!RHOKS = -1.       $ CONSTRAINT LUMPING PARAMETER (<0 don't lump)
!KPEN = 6000.      $ LP PENALTY COEFF.
!MMAJOR = 10       $ MAXIMUM NUMBER OF MAJOR ITERATIONS
!MMINOR = 4        $ MAXIMUM NUMBER OF MINOR ITERATIONS
!NLST=1            $ NUMBER OF LOAD SETS (USES DIFF BC OR TEMPERATURES)
!NUMD=1            $ NUMBER OF DISPLACEMENT CONTRAINTS
!BLEN=20           $ DEFAULT CHARACTERISTIC BUCKLING LENGTH
*TF OPEN 19 'opt_output     $ assign file for optimizer output
$ assign various work libraries for use by eal - should not need mods
!ALIB=6       $ A SCRATCH AUS LIBRARY
!SLIB=9       $ LIB FOR SAVED DESIGN VARIABLES, STRESS RESULTANTS, AND DERIVs
!OTIB=15      $ OUTPUT LIBRARY FOR INTERMEDIATE STRESS OR DISP. DERIVATIVES
!OLIB=15
!LLIB=4       $ LIBRARY FOR CHARACTERISTIC BUCKLING LENGTHS
*ECHO 4
$ EXITING: (SET PARA) $ SET CONTROL PARAMETERS
*NOECHO 4
*RETURN
*                    ENDPAR
```

Optimization Problem Definition Datasets

Data tables which complete the description of the optimization problem are contained in the procedure DESV DEFN. The COMP GRP and COMP DV tables relate the design variables to the appropriate element groups and assign section and material types. The first dataset required is the COMP GRP table. Detailed descriptions of the tables in the DESV DEFN procedure are contained in section 2.1, so only a summary is given here. The COMP GRP dataset relates groups of elements to design components. A component is defined as a group of design variables which together completely describe a section type. There may be more than one element group related to the same component. Since each element is being designed independently in this example, each has its own component. Note that element nine has material 2 assigned to it. The different stress limits for materials 1 and 2 will be seen later in the matprop.prn file.

The next important dataset is the COMP DV table. This table contains the design variables, the component each variable is associated with, and the upper and lower design variable bounds. Looking at the COMP DV dataset, each component has a single design variable whose initial value is 5.0. The minimum area constraint is specified as 0.1 in column 4 of the table, and the scale factor is 1.0 (column 7). This dataset will be updated with the current design variables as the optimization proceeds. A current copy of this table is also copied to Library 3 after each iteration so that a complete design history is stored there. The COMP DV dataset stored in Library 3 is also used for restarts. Upon execution, the procedure checks Library 3 to see if a COMP DV dataset is present and uses it if it exists.

The E23 LENG ngrp datasets are shown as examples for four groups. These do not affect the results since there is no local buckling check for rod elements, and they are included only to show the table format. These tables can exist for certain element groups and be omitted for others. A DEFL DEFI dataset must be defined for each problem whether displacement constraints are called for or not. In this case, a displacement limit of 1000 in. was defined for node 6 in the two direction. Since this node is constrained by a boundary condition, the displacement constraint will have no effect. The final dataset required is the LSET CASE table. This table defines that there is one load case in the one load set.

Optimization Problem Definition Procedure (DESV DEFN)

```
*(DESV DFN)          ENDDVD
*ECHO 4
$ ENTERING: (DESV DFN)
$ COMPONENT AND DESIGN VARIABLE DEFINITION
*NOECHO 4
$
*XQT AUS
  TABLE (NI=4,NJ=10,TYPE=0): COMP GRP
$  IN TABLE COMP GROUP NI=4, NJ=TOTAL NUMBER OF ELEMENT GROUPS
$   I=1;  COMPONENT NUMBER - FOR ASSOCIATION IN COMP DV
$   I=2;  MATERIAL AND SECTION TYPE INDICATOR
$   I=3;  ELEMENT TYPE (E21,E43,ETC)
$   I=4;  ELEMENT GROUP NUMBER
  I=1,2,3,4
 J=1: 1 1 E23 1
 J=2: 2 1 E23 2
 J=3: 3 1 E23 3
 J=4: 4 1 E23 4
 J=5: 5 1 E23 5
 J=6: 6 1 E23 6
 J=7: 7 1 E23 7
 J=8: 8 1 E23 8
 J=9: 9 2 E23 9          $element 9 uses material 2
 J=10: 10 1 E23 10
```

```
$
TABLE(NI=8,NJ=10):COMP DV
$ I=1: Component Number - correspondence with comp grp
$ I=2: Design variable number in component
$ I=3: Design variable value
$ I=4: Lower bound on design variable
$ I=5: Upper bound on design variable
$ I=6: Constant C0 in linear formulation of val=C0+C1*DV
$ I=7: Coefficient C1 in linear formulation of val=C0+C1*DV
$ I=8: Status flag of design variable (controlled by program)
 J=  1 : 1. 1.0    5.0   0.1     42.    0.    1.  0.
 J=  2 : 2. 1.0    5.0   0.1     42.    0.    1.  0.
 J=  3 : 3. 1.0    5.0   0.1     42.    0.    1.  0.
 J=  4 : 4. 1.0    5.0   0.1     42.    0.    1.  0.
 J=  5 : 5. 1.0    5.0   0.1     42.    0.    1.  0.
 J=  6 : 6. 1.0    5.0   0.1     42.    0.    1.  0.
 J=  7 : 7. 1.0    5.0   0.1     42.    0.    1.  0.
 J=  8 : 8. 1.0    5.0   0.1     42.    0.    1.  0.
 J=  9 : 9. 1.0    5.0   0.1     42.    0.    1.  0.
 J= 10: 10. 1.0    5.0   0.1     42.    0.    1.  0.
$
*XQT DCU
$ COPY COMP DV FROM LIB 3 IF IT EXISTS (RESTART)
 COPY 3 1 COMP DV 1 1  $ get COMP DV from LIB 3 if one exists
*XQT AUS
 OUTLIB="LLIB"
TABLE(NI=1,NJ=1): E23 LENG 1
  J=1 : 24.
TABLE(NI=1,NJ=1): E23 LENG 2
  J=1 : 24.
TABLE(NI=1,NJ=1): E23 LENG 7
  J=1 : 24.
TABLE(NI=1,NJ=1): E23 LENG 9
  J=1 : 24.
OUTLIB=1
TABLE(NI=3,NJ="NUMD"):DEFL DEFI
  J=1: 6. 2. 1000.      $ dummy constraint
TABLE(NI=2,NJ="NLST",TYPE=0): LSET CASE
   I=1              $NUMBER OF LOAD CASES IN SET J
   J=1:   1         $LOADSET 1 HAS 1 LOADCASE
$
*ECHO 4
$ EXITING: (DESV DFN) $ COMPONENT AND DESIGN VARIABLE DEFINITION
*NOECHO 4
$
*RETURN
*                   ENDDVD
```

Materials Data File (matprop.prn)

The material data file used for the ten-bar truss problem is shown below. Note that material 2 used for bar 9 has the higher allowable stress. For the bar element, the Poisson's ratio and shear properties have no effect and may be omitted. The matprop.prn file must be in the directory where the optimization is being run.

```
   1            bar test problem properties 25KSI strength
60.0           10000000.      10000000.       3850000.        .31
.31            .10            25000.           0.              0.
0.             25000.          0.              0.              0.
10000.         0.              0.              0.              0.
0.             0.              0.              0.              0.
0.             0.              0.              0.              0.
0.             0.              0.              0.              0.
0.             0.              0.              0.              0.
.0000048       .0000048        0.              0.              0.
   2           bar test problem properties 75KSI strength
60.0           10000000.      10000000.       3850000.        .31
.31            .10            75000.           0.              0.
0.             75000.          0.              0.              0.
30000.         0.              0.              0.              0.
0.             0.              0.              0.              0.
0.             0.              0.              0.              0.
0.             0.              0.              0.              0.
0.             0.              0.              0.              0.
.0000048       .0000048        0.              0.              0.
```

3.2 Output from the Linear Programming Processor LIP4

After assembling the optimization problem and executing EAL, the next step is interpreting the results. The results of greatest interest will be in a file containing the output from the LIP4 processor - the linear programming results for each iteration. The following listing is the output from the first several iterations from the ten-bar truss problem solution. Additional comments are denoted by bold type.

```
 REG MAJO=              1                    First major iteration
 REG MINO=              0                    No minor iterations performed
 REG NEWP= 0.1087468399495650E+05            Value of computed penalty
 REG ML  = 0.4000000000000000E+00            40% move limits used
 EXIT AUS          4.890     2198     5032      981     2233
 LPEN=       1
 KPEN  0.60000000E+04                        Penalty coefficient=6000
 LIP4 330.06   DataSpace= 1000000, Date/Time= 951024 183358
  MINOS ERROR FLAG =            0
  NEW OBJECTIVE FUNCTION =    1843.785900431802   Objective after LP cycle
   CHANGE FROM LP =   -254.4478957429461            Change in Objective
     PENALTIES USED
        OBJECTIVE =   0.0000000000000000E+00        No penalty required since all
  (OBJ COEFFICIENT=   0.0000000000000000E+00 )      constraints OK after LP
```

The following section shows the values of the design variables before and after the LP cycle. In general, the bounds are a combination of move limits and variable bounds defined in COMP DV.

DESIGN VARIABLES

DV #	LOWER BOUND	INIT DV	DV	UPPER BOUND	BOUND
1	0.30000E+01	0.50000E+01	0.70000E+01	0.70000E+01	UPPR
2	0.30000E+01	0.50000E+01	0.30000E+01	0.70000E+01	LOWR
3	0.30000E+01	0.50000E+01	0.70000E+01	0.70000E+01	UPPR
4	0.30000E+01	0.50000E+01	0.30000E+01	0.70000E+01	LOWR
5	0.30000E+01	0.50000E+01	0.30000E+01	0.70000E+01	LOWR
6	0.30000E+01	0.50000E+01	0.30000E+01	0.70000E+01	LOWR
7	0.30000E+01	0.50000E+01	0.58824E+01	0.70000E+01	FREE
8	0.30000E+01	0.50000E+01	0.59482E+01	0.70000E+01	FREE
9	0.30000E+01	0.50000E+01	0.30000E+01	0.70000E+01	LOWR
10	0.30000E+01	0.50000E+01	0.30000E+01	0.70000E+01	LOWR

This section shows the value of the constraints before and after the LP cycle. The constraints are ordered in pairs, the first being the stress and the second being local buckling. The order of the pairs will be by group appearance in the COMP GRP dataset. If rhoks<0, constraint values will be printed for each element. If rhoks>0, a single pair of K-S function constraints will be printed for each group. After all stress and buckling constraints are printed, displacement constraints are listed (#21 in this example). The buckling constraints are all -1 for the rod elements in this example. The Lagrange multiplier gives the relative importance of each constraint and should be used to verify that an appropriate value of KPEN was used.

CONSTRAINTS

CON #	SIGCT	INIT CON	CON	LAGRANGE MULT
1	10	0.56292E+00	-.80000E-01	0.89866E-13
2	10	-.10000E+01	-.10000E+01	0.00000E+00
3	10	-.67900E+00	-.54877E+00	0.00000E+00
4	10	-.10000E+01	-.10000E+01	0.00000E+00
5	10	0.63708E+00	-.34694E-16	-.37125E+03

```
     6     10 -.10000E+01          -.10000E+01          0.00000E+00
     7     10 -.52100E+00          -.33123E+00          0.00000E+00
     8     10 -.10000E+01          -.10000E+01          0.00000E+00
     9     10 -.71608E+00          -.61843E+00          0.00000E+00
    10     10 -.10000E+01          -.10000E+01          0.00000E+00
    11     10 -.67900E+00          -.54877E+00          0.00000E+00
    12     10 -.10000E+01          -.10000E+01          0.00000E+00
    13     10 0.18381E+00          0.44235E-16          -.45097E+03
    14     10 -.10000E+01          -.10000E+01          0.00000E+00
    15     10 0.78932E-01          -.15079E+00          0.00000E+00
    16     10 -.10000E+01          -.10000E+01          0.00000E+00
    17     10 -.77420E+00          -.68474E+00          0.00000E+00
    18     10 -.10000E+01          -.10000E+01          0.00000E+00
    19     10 -.54604E+00          -.36186E+00          0.00000E+00
    20     10 -.10000E+01          -.10000E+01          0.00000E+00
    21     10 -.10000E+01          -.10000E+01          0.00000E+00
 EXIT LIP4         4.910     2204     5083      982     2257
 REG MAJO=             1            Still first major iteration
 REG MINO=             1            This is a minor (approx sens. calc)
 REG NEWP= 0.3802952301922171E+04
 REG ML  = 0.4000000000000000E+00
 EXIT AUS          5.400     2544     5798     1124     2547
 LPEN=       1
 KPEN  0.60000000E+04
 LIP4 330.06   DataSpace= 1000000, Date/Time= 951024 183358
  MINOS ERROR FLAG =            0
  NEW OBJECTIVE FUNCTION =    1628.556803223251
   CHANGE FROM LP =   -215.2290937065959
     PENALTIES USED
        OBJECTIVE =   0.0000000000000000E+00
  (OBJ COEFFICIENT=   0.0000000000000000E+00 )


  DESIGN VARIABLES
DV #  LOWER BOUND          INIT DV          DV          UPPER BOUND      BOUND
   1  0.42000E+01      0.70000E+01      0.98000E+01      0.98000E+01      UPPR
   2  0.18000E+01      0.30000E+01      0.18000E+01      0.42000E+01      LOWR
   3  0.42000E+01      0.70000E+01      0.62301E+01      0.98000E+01      FREE
   4  0.18000E+01      0.30000E+01      0.22665E+01      0.42000E+01      FREE
   5  0.18000E+01      0.30000E+01      0.18000E+01      0.42000E+01      LOWR
   6  0.18000E+01      0.30000E+01      0.18000E+01      0.42000E+01      LOWR
   7  0.35294E+01      0.58824E+01      0.40582E+01      0.82353E+01      FREE
   8  0.35689E+01      0.59482E+01      0.75737E+01      0.83275E+01      FREE
   9  0.18000E+01      0.30000E+01      0.18000E+01      0.42000E+01      LOWR
  10  0.18000E+01      0.30000E+01      0.18000E+01      0.42000E+01      LOWR

CONSTRAINTS
CON #   SIGCT    INIT CON              CON              LAGRANGE MULT
     1     10 0.95823E-01          -.70690E-16          -.27367E+03
     2     10 -.10000E+01          -.10000E+01          0.00000E+00
     3     10 -.46004E+00          -.31442E+00          0.00000E+00
     4     10 -.10000E+01          -.10000E+01          0.00000E+00
     5     10 0.18989E+00          -.71557E-16          -.60900E+03
     6     10 -.10000E+01          -.10000E+01          0.00000E+00
     7     10 -.20663E+00          0.41633E-16          -.18104E+03
     8     10 -.10000E+01          -.10000E+01          0.00000E+00
     9     10 -.56979E+00          -.18904E+00          0.00000E+00
    10     10 -.10000E+01          -.10000E+01          0.00000E+00
```

```
    11     10 -.46004E+00          -.31442E+00          0.00000E+00
    12     10 -.10000E+01          -.10000E+01          0.00000E+00
    13     10 0.40813E-01          0.11015E-15          -.59284E+03
    14     10 -.10000E+01          -.10000E+01          0.00000E+00
    15     10 -.12726E+00          -.98409E-02          0.35194E-12
    16     10 -.10000E+01          -.10000E+01          0.00000E+00
    17     10 -.62600E+00          -.48499E+00          0.00000E+00
    18     10 -.10000E+01          -.10000E+01          0.00000E+00
    19     10 -.23638E+00          -.31465E-01          0.00000E+00
    20     10 -.10000E+01          -.10000E+01          0.00000E+00
    21     10 -.10000E+01          -.10000E+01          0.00000E+00
 EXIT LIP4         5.430     2550     5846     1125     2568
```

After this LP execution a warning is printed that the penalty value increased after the last minor iteration indicating the approximate derivatives may be of poor quality. Another possibility is that the value of the penalty parameter KPEN is too large. KPEN is set at 6000 and the largest Lagrange multiplier is about 600. These results will be discarded and a new, full sensitivity calculation (major iteration) will be performed.

```
 $ WARNING!!! PENALTY INCREASED DURING LAST OPTIMIZATION CYCLE

 $ THIS CYCLE WILL BE DISCARDED AND THE DESIGN VARIABLES RESET

 REG OLDP= 0.3802952301922171E+04       $ penalty from previous iteration
 REG NEWP= 0.6420127158036401E+04       $ penalty from latest iteration
 REG MAJO=              2               $ Now on second major iteration
 REG MINO=              0
 REG NEWP= 0.3802952301922171E+04
 REG ML  = 0.4000000000000000E+00
 EXIT AUS          9.250     4753    11212     2115     4951
 LPEN=       1
 KPEN  0.60000000E+04
 LIP4 330.06   DataSpace= 1000000, Date/Time= 951024 183402
  MINOS ERROR FLAG =            0
  NEW OBJECTIVE FUNCTION =    1636.033455000091
   CHANGE FROM LP =   -207.7524419297553
     PENALTIES USED
        OBJECTIVE =   0.0000000000000000E+00
  (OBJ COEFFICIENT=   0.0000000000000000E+00 )
```

note that the initial design variables are from last acceptable cycle

```
DESIGN VARIABLES
DV #  LOWER BOUND        INIT DV          DV          UPPER BOUND     BOUND
   1  0.42000E+01     0.70000E+01     0.63365E+01     0.98000E+01      FREE
   2  0.18000E+01     0.30000E+01     0.18000E+01     0.42000E+01      LOWR
   3  0.42000E+01     0.70000E+01     0.92919E+01     0.98000E+01      FREE
   4  0.18000E+01     0.30000E+01     0.18986E+01     0.42000E+01      FREE
   5  0.18000E+01     0.30000E+01     0.18000E+01     0.42000E+01      LOWR
   6  0.18000E+01     0.30000E+01     0.18000E+01     0.42000E+01      LOWR
   7  0.35294E+01     0.58824E+01     0.80124E+01     0.82353E+01      FREE
   8  0.35689E+01     0.59482E+01     0.35689E+01     0.83275E+01      LOWR
   9  0.18000E+01     0.30000E+01     0.18000E+01     0.42000E+01      LOWR
  10  0.18000E+01     0.30000E+01     0.25416E+01     0.42000E+01      FREE
```

```
CONSTRAINTS
CON #   SIGCT    INIT CON              CON              LAGRANGE MULT
    1      10 0.95823E-01          0.42501E-16          -.13022E+03
    2      10 -.10000E+01          -.10000E+01          0.00000E+00
    3      10 -.46004E+00          -.15941E+00          0.00000E+00
    4      10 -.10000E+01          -.10000E+01          0.00000E+00
    5      10 0.18989E+00          -.30791E-16          -.31153E+03
    6      10 -.10000E+01          -.10000E+01          0.00000E+00
    7      10 -.20663E+00          -.24286E-16          -.93839E+02
    8      10 -.10000E+01          -.10000E+01          0.00000E+00
    9      10 -.56979E+00          -.77899E+00          0.00000E+00
   10      10 -.10000E+01          -.10000E+01          0.00000E+00
   11      10 -.46004E+00          -.15941E+00          0.00000E+00
   12      10 -.10000E+01          -.10000E+01          0.00000E+00
   13      10 0.40813E-01          -.13010E-17          -.48722E+03
   14      10 -.10000E+01          -.10000E+01          0.00000E+00
   15      10 -.12726E+00          -.11050E+00          0.00000E+00
   16      10 -.10000E+01          -.10000E+01          0.00000E+00
   17      10 -.62600E+00          -.51630E+00          0.00000E+00
   18      10 -.10000E+01          -.10000E+01          0.00000E+00
   19      10 -.23638E+00          0.00000E+00          -.28459E+03
   20      10 -.10000E+01          -.10000E+01          0.00000E+00
   21      10 -.10000E+01          -.10000E+01          0.00000E+00
 EXIT LIP4         9.260     4759    11260     2116     4972
```

3.3 Restarts and Parameter Settings

This section describes how to use the simple restart procedure from the latest design and also provides insight on setting the various optimization parameters. The ability to restart the optimization procedure from any design step is an important feature. Each intermediate design is stored in Library 3 (the L03 file) during program execution. During the DESV DEFN procedure, there is a command to copy any active COMP DV dataset from Library 3. The design variable values from this table, if present, will be used. An intermediate design variable set may be selected by enabling the appropriate COMP DV dataset using EAL processor DCU commands. A listing of the table of contents in Library 3 will show each intermediate COMP DV dataset. The procedure for restarting the optimization from the last design cycle is: 1) make necessary modifications to any of the optimization parameters in SET PARA, 2) delete all previous EAL libraries except Library 3, and 3) resubmit the EAL optimization runstream.

Several of the optimization control parameters, initialized in the procedure SET PARA, require somewhat arbitrary definition by the user. In this section, some strategies for selecting values which will speed execution time and provide satisfactory solutions will be discussed. Proper settings for these parameters are problem dependent. It is up to the user to monitor the solution progress and modify the various parameters as required. The following discussion is meant to aid in that process.

MMAJOR It is suggested that the number of major iterations be limited to two or three for the first pass at the optimization. This will give the user the opportunity to evaluate the solution progress and the various parameter settings. Since there is no convergence criterion, the procedure will continue until the major iteration limit is reached. It is suggested that a number of major iterations be selected for subsequent restarts with computer time requirements considered.

MMINOR Experience has shown that minor iterations are most valuable for large problems where a full sensitivity calculation is very expensive. They also seem to most helpful during the initial iterations when constraints may be far from their boundaries and the quality of the sensitivity derivatives is not as critical. Since minor iterations are relatively cheap in terms of computer time, it is suggested that a relatively large number of minor iterations be allowed in the early iterations. Since the minor iteration cycle is stopped when the penalized objective function increases, there is little danger of the minor

iterations leading to bad solutions. The user should monitor the output, and if none of the minor iteration cycles is accepted then the MMINOR parameter should be set to zero.

ML The selection of the move limit factor is somewhat dependent on how close the current design is to the final solution. If the design is far away from the final design, than large move limits and approximate sensitivity calculations can speed the process immensely. As the final design is approached, large move limits can result in oscillation. Methodology is in place to reduce the move limit as the penalized objective function increases. The execution time penalty for each move limit reduction is an execution of the linear programming processor and a static analysis. This time is generally small compared to a full sensitivity calculation, so it is recommended that relatively large move limits (0.25-0.50) be used for initial executions. For restarts, it is generally a good idea to look at the move limits and solution behavior of the previous iteration.

RHOKS The first consideration in setting the rho parameter for the K-S function is deciding if the constraints should be lumped or not. For small problems, such as the ten-bar truss, there is really no reason to lump constraints, so a negative value should be used for RHOKS. For large problems, memory limitations encountered in the LIP4 processor may necessitate constraint lumping, and constraint lumping can ease result interpretation for many mid-size problems. Typical values of RHOKS when constraint lumping is used range from 10 to 300, and a value of 50 has generally provided good results.

DR The finite difference step size chosen can have an affect on the quality of the sensitivities. It is suggested that the user try several values of DR for a few iterations to see if this affects the results. Generally acceptable results have been obtained for DR in the range of 10^{-4} to 10^{-7}.

KPEN The penalty parameter KPEN is probably the most difficult parameter to select prior to an optimization run since acceptable values are problem dependent. Most importantly, if the value of KPEN is too small the constraints will not be satisfied. The best way to choose a value of KPEN is to examine the Lagrange multipliers from the linear programming output for an iteration when all the final linearized constraints are satisfied. A value of KPEN that is a few times larger than the largest absolute value of the Lagrange multipliers has been found to be satisfactory. If there are violated constraints, it is important to observe the progress of the optimization and make sure KPEN is large enough so that the values of the violated constraints are reduced during each iteration. The danger in setting an arbitrarily large value of KPEN is that progress to the final solution will be slowed.

CONCLUDING REMARKS

This report has presented a structural optimization procedure developed for use with the Engineering Analysis Language (EAL) finite element analysis code. The procedure is written primarily in the EAL command language and uses three external FORTRAN processors. The first section of the report describes the methodology and flow of the procedure, and the second section describes the optimization problem inputs and section models. The final section uses the ten-bar truss problem to further describe the procedures' input and outputs. The appendix contains a listing, with comments, of the optimization procedure.

The procedure described in this report has been used successfully on a number of large and small design and optimization projects at the NASA Langley Research Center. An emphasis was placed on producing efficient methods and implementations for use with large optimization and finite element models. The use of partial stiffness matrices in the sensitivity calculations and the approximate updates of constraint sensitivities can greatly reduce computer time and storage requirements. The performance improvements attained using the approximate sensitivity updates are described in reference 3 for optimization problems of various sizes. In addition, the capability to include built-up structural sections with realistic stiffnesses and constraints adds to the procedure's utility as an aircraft design tool. The stiffnesses and constraints for the built-up sections are computed in external processors which are written in FORTRAN and are linked with the EAL analysis code. The use of FORTRAN external processors allows for the writing of complex constraint and stiffness formulations and makes the addition of section geometries relatively straightforward.

REFERENCES

1. Whetstone, W. D.: EISI-EAL Engineering Analysis Reference Manual. Engineering Information Systems, Inc., July 1983.

2. Murtagh, Bruce A. and Saunders, Michael A.: MINOS 5.4 User's Guide (preliminary). Technical Report SOL 83-20R, Systems Optimization Laboratory, Stanford University. Stanford, CA, 1983 (revised 1993).

3. Scotti, Steven J.: Structural Design Using Equilibrium Programming Formulations. NASA TM-110175, June 1995.

4. Camarda, Charles J. and Adelman, Howard M.: Static and Dynamic Structural-Sensitivity Derivative Calculations in the Finite-Element-Based Engineering Analysis Language (EAL) System. NASA TM-85734, 1984.

5. Zhang, Nae-Heon, and Lasdon; "An Improved Successive Linear Programming Algorithm", Management Science, Vol. 31. No. 10, October 1985, pp. 1312-1331.

6. Walsh, Joanne L.: Application of Mathematical Optimization Procedures to a Structural Model of a Large Finite-Element Wing. NASA TM-87597, 1986.

7. Haftka, R. T., Zafer, G., and Kamat, M. P.: Elements of Structural Optimization. Kluwer Academic Publishers, Boston, MA, 1990.

8. Giles, Gary L. and Haftka, Raphael T.: SPAR Data Handling Utilities. NASA TM-78701, 1978.

9. Cunningham, Sally W.: SPAR Data Set Contents. NASA TM-83181, 1981.

APPENDIX: Listing of Optimization Runstream

```
$ EAL STRUCTURAL OPTIMIZATION RUNSTREAM
*ONLINE = 0
*NOECHO 1,2,3,4
*XQT U1
$
$-------------------------------------------------------------------------*
$
*(INIT MODL)       ENDINM
*ECHO 4
$ ENTERING: (INIT MODL)
$ INITIALIZE EAL MODEL
*NOECHO 4
*XQT TAB   $ ten bar truss
  START 6 3 4 5 6
  MATC: 1 1.+7 .3 .1
  BC: 1 5.
  JLOC
  1 720. 360.
  2 720. 0.
  3 360. 360.
  4 360. 0.
  5 0.   360.
  6 0.   0.
 CON=1
  ZERO 1 2: 5,6
*XQT ELD
 E23
  GROUP 1:5 3
  GROUP 2:3 1
  GROUP 3:6 4
  GROUP 4:4 2
  GROUP 5:3 4
  GROUP 6:1 2
  GROUP 7:5 4
  GROUP 8:6 3
  GROUP 9:3 2
  GROUP 10:4 1
*XQT AUS
  SYSVEC: APPL FORC : I=2
   J=2: -100000.
   J=4: -100000.
*XQT E
*XQT TAN
$
*ECHO 4
$ EXITING: (INIT MODL) $ INITIALIZE EAL MODEL
*NOECHO 4
$
*RETURN
*              ENDINM
$
$-------------------------------------------------------------------------*
$
*(SET PARA)       ENDPAR
*ECHO 4
$ ENTERING: (SET PARA)
```

```
$ SET CONTROL PARAMETERS
*NOECHO 4
$
!NDV= 10        $ NO.OF INDEPENDENT DESIGN VARIABLES
!DR=.1E-7       $ FACTOR FOR INCREMENTING DESIGN VARIABLE
!ML=.30         $ MOVE LIMIT ON DESIGN VARIABLES
!RHOKS = -1.    $ CONSTRAINT LUMPING PARAMETER (<0 don't lump)
!KPEN = 500.    $ LP PENALTY COEFF.
!MMAJOR =12     $ MAXIMUM NUMBER OF MAJOR ITERATIONS
!MMINOR = 2     $ MAXIMUM NUMBER OF MINOR ITERATIONS
!NLST=1         $ NUMBER OF LOAD SETS (USES DIFF BC OR TEMPERATURES)
!NUMD=1         $ NUMBER OF DISPLACEMENT CONTRAINTS
!BLEN=20.       $ DEFAULT CHARACTERISTIC BUCKLING LENGTH USED BY CONS
!GLIM=-.5
*TF OPEN 19 'optimization_results
$ assign various work libraries for use by eal - should not need to be modified
!ALIB=6     $ A SCRATCH AUS LIBRARY
!SLIB=9     $ LIB FOR SAVED DESIGN VARIABLES, STRESS RESULTANTS, AND DERIVATIVE
!OTIB=15    $ OUTPUT LIBRARY FOR INTERMEDIATE STRESS OR DISP. DERIVATIVES
!OLIB=15
!LLIB=4     $ LIBRARY FOR CHARACTERISTIC BUCKLING LENGTHS
*ECHO 4
$ EXITING: (SET PARA) $ SET CONTROL PARAMETERS
*NOECHO 4
$
*RETURN
*               ENDPAR
$
$-------------------------------------------------------------------------------*
$
*(DESV DFN)       ENDDVD
*ECHO 4
$ ENTERING: (DESV DFN)
$ COMPONENT AND DESIGN VARIABLE DEFINITION
*NOECHO 4
$
*XQT AUS
$
$$        $$ DESIGN VARIABLE DEFINITION
 TABLE (NI=4,NJ=10,TYPE=0): COMP GRP
$  IN TABLE COMP GROUP NI=4, NJ=TOTAL NUMBER OF ELEMENT GROUPS
$   I=1;  COMPONENT NUMBER - FOR ASSOCIATION IN COMP DV
$   I=2;  MATERIAL AND SECTION TYPE INDICATOR
$   I=3;  ELEMENT TYPE (E21,E43,ETC)
$   I=4;  ELEMENT GROUP NUMBER
 I=1,2,3,4
 J=1: 1 1 E23 1
 J=2: 2 1 E23 2
 J=3: 3 1 E23 3
 J=4: 4 1 E23 4
 J=5: 5 1 E23 5
 J=6: 6 1 E23 6
 J=7: 7 1 E23 7
 J=8: 8 1 E23 8
 J=9: 9 2 E23 9
 J=10: 10 1 E23 10
$
TABLE(NI=8,NJ=10):COMP DV
```

```
$ I=1: Component Number - correspondence with comp grp
$ I=2: Design variable number in component
$ I=3: Design variable value
$ I=4: Lower bound on design variable
$ I=5: Upper bound on design variable
$ I=6: Constant C0 in linear formulation of val=C0+C1*DV
$ I=7: Coefficient C1 in linear formulation of val=C0+C1*DV
$ I=8: Status flag of design variable (controlled by program)
 J= 1 : 1. 1.0   5.0  0.1    42.   0.   1. 0.
 J= 2 : 2. 1.0   5.0  0.1    42.   0.   1. 0.
 J= 3 : 3. 1.0   5.0  0.1    42.   0.   1. 0.
 J= 4 : 4. 1.0   5.0  0.1    42.   0.   1. 0.
 J= 5 : 5. 1.0   5.0  0.1    42.   0.   1. 0.
 J= 6 : 6. 1.0   5.0  0.1    42.   0.   1. 0.
 J= 7 : 7. 1.0   5.0  0.1    42.   0.   1. 0.
 J= 8 : 8. 1.0   5.0  0.1    42.   0.   1. 0.
 J= 9 : 9. 1.0   5.0  0.1    42.   0.   1. 0.
 J= 10: 10. 1.0   5.0  0.1    42.   0.   1. 0.
$
*XQT DCU
 COPY 3 1 COMP DV 1 1  $ get COMP DV from LIB 3 if one exists
*XQT AUS
 OUTLIB="LLIB"
TABLE(NI=1,NJ=1): E23 LENG 1
 J=1 : 24.
TABLE(NI=1,NJ=1): E23 LENG 2
 J=1 : 24.
TABLE(NI=1,NJ=1): E23 LENG 3
 J=1 : 24.
TABLE(NI=1,NJ=1): E23 LENG 4
 J=1 : 24.
TABLE(NI=1,NJ=1): E23 LENG 5
 J=1 : 24.
OUTLIB=1
TABLE(NI=3,NJ="NUMD"):DEFL DEFI
 J=1: 6. 2.  100.
TABLE(NI=2,NJ="NLST",TYPE=0): LSET CASE
  I=1: J=1: 1
$
*ECHO 4
$ EXITING: (DESV DFN) $ COMPONENT AND DESIGN VARIABLE DEFINITION
*NOECHO 4
$
*RETURN
*             ENDDVD
$
$-------------------------------------------------------------------------*
$
*(DRIV OPT)       ENDOPT
*ECHO 4
$ ENTERING: (DRIV OPT)
$ DRIVER PROGRAM TO OPTIMIZE A STRUCTURE SUBJECT TO STRESS AND DISP CONSTRAINTS
*NOECHO 4
$
*DCALL(INIT MODL)     $ INITIALIZE EAL MODEL
*DCALL(SET PARA)      $ SET CONTROL PARAMETERS
*DCALL(DESV DFN)      $ DEFINE COMPONENTS AND D.V.'S
*DCALL(ADS INIT)      $ INITIALIZE DATA SET REQUIRED BY ADS
```

```
$ initialize counters
!EXACT = 1
!MAJORITER = 0
!MINORITER = 0
!FIRST = 1
!ICT = 0
$
*LABEL BMAJORITER      $ start of major iteration loop
*DCALL(CLEA NL01)      $ CLEAR UP LIBRARIES
!NNST = 1              $ NOTE: NLST = NUMBER OF LOAD SETS
*LABEL BLOADSET
*IF("NNST" GT "NLST"): *GO TO ELOADSET        $ LOOP OVER LOAD SETS
  !LCAS=DS,1,"NNST",1(1,LSET,CASE,MASK,MASK)
  !OLIB = 1
*IF("GLIM" GE 0.0): *GO TO EGCOPY
*XQT DCU
  COPY  "SLIB" 1 GACT E21
  COPY  "SLIB" 1 GACT E23
  COPY  "SLIB" 1 GACT E31
  COPY  "SLIB" 1 GACT E32
  COPY  "SLIB" 1 GACT E33
  COPY  "SLIB" 1 GACT E41
  COPY  "SLIB" 1 GACT E42
  COPY  "SLIB" 1 GACT E43
  COPY  "SLIB" 1 GACT E44
*LABEL EGCOPY
*  DCALL(CALC STAT)    $ PERFORM STATIC ANALYSIS
*  XQT CONS            $ PROCESSOR TO EVALUATE CONSTRAINTS, UNPERTURBED DESIGN
    RESET RHOK = "RHOKS" SET="NNST" NOUT="OLIB" BLEN="BLEN"  LLIB="LLIB"
    RESET NOJUNK=YES
  !STATUS = 1
*  DCALL(DISP CONS)    $ COMPUTE DISPLACEMENT CONSTRAINT
  !NNST = NNST + 1
*  GO TO BLOADSET       $ repeat for each load set
*LABEL ELOADSET
!STATUS = 1
*DCALL(ADS CONS)       $ WRITE OUT CONSTRAINTS IN ADS FORMAT
$
$ CALC PENALTY HERE, IF MINORITER ZERO, GO ON TO EXACT SENS
$ IF MINORITER NOT ZERO AND PENALTY REDUCED DO MINOR,
$ IF NOT RESET X ADS AND DOCALC STAT AND EXACT SENS
*DCALL(CALC PEN)
$ if first time thru or number of minor iters exceeds limit don't update cons
*IF("FIRST" NE 0):*GO TO NOUP
$ if penalty function has decreased than do another approximate update
*IF("NEWPEN" LT "OLDPEN"):*GO TO DOUP
*TF STANDARD OUTPUT=19  $redirect optimizer output
*ECHO 4
$ WARNING!!! PENALTY INCREASED DURING LAST OPTIMIZATION CYCLE
$ THIS CYCLE WILL BE DISCARDED AND THE DESIGN VARIABLES RESET
*NOECHO 4
!OLDPEN: !NEWPEN
*TF STANDARD OUTPUT=6
!BADPEN = NEWPEN
!NEWPEN = OLDPEN
$
*IF("EXACT" EQ 0):*GOTO SKIP
$ IF LAST COMPUTATION WAS EXACT AND PENALTY RISING
```

```
  !ML = ABS(ML)/2.0  $ REDUCE MOVE LIMITS
$ RESET DESIGN VARIABLES AND REPEAT OPTIMIZATION
*XQT AUS
 DEFINE X = "SLIB" X ADS 1 1
 TABLE,U(TYPE=-1): 1 X ADS 1 1
  TRANSFER(SOURCE = X, OPER=XSUM)
*XQT AUS
 DEFINE F = "SLIB" F ADS 1 1
 TABLE,U(TYPE=-1): 1 F ADS 1 1
  TRANSFER(SOURCE = F, OPER=XSUM)
*XQT AUS
 DEFINE G = "SLIB" G ADS 1 1
 TABLE,U(TYPE=-1): 1 G ADS 1 1
  TRANSFER(SOURCE = G, OPER=XSUM)
*  GO TO DOOPT      $ REPEAT OPTIMIZATION WITH REDUCED MOVE LIMITS
$
*LABEL SKIP
$ THIS LEAVES THE CASE WHERE PENALTY ROSE AFTER APPROXIMATE ANALYSIS
$ RESET DESIGN VARIABLES AND RECALC STATIC
*XQT AUS
 DEFINE X = "SLIB" X ADS 1 1
 TABLE,U(TYPE=-1): 1 X ADS 1 1
  TRANSFER(SOURCE = X, OPER=XSUM)
*DCALL(CLEA NL01)      $ CLEAR UP LIBRARIES
!NNST = 1           $ NOTE: NLST = NUMBER OF LOAD SETS
*LABEL B1LOADSET
*IF("NNST" GT "NLST"): *GO TO E1LOADSET        $ LOOP OVER LOAD SETS
  !LCAS=DS,1,"NNST",1(1,LSET,CASE,MASK,MASK)
  !OLIB = 1
*  DCALL(CALC STAT)      $ PERFORM STATIC ANALYSIS
*  XQT CONS    $ PROCESSOR TO EVALUATE CONSTRAINTS, UNPERTURBED DESIGN
   RESET RHOK = "RHOKS" SET="NNST" NOUT="OLIB" BLEN="BLEN" LLIB="LLIB"
   RESET NOJUNK=YES
  !STATUS = 1
*  DCALL(DISP CONS)       $ COMPUTE DISPLACEMENT CONSTRAINT
!NNST = NNST + 1
*GO TO B1LOADSET
*LABEL E1LOADSET
!STATUS = 1
*DCALL(ADS CONS)       $ WRITE OUT CONSTRAINTS IN ADS FORMAT
*GOTO NOUP
$
*LABEL DOUP
$  UPDATE CONSTRAINT GRADIENTS
*IF("MINORITER" GE "MMINOR"): *GOTO NOUP $ UNLESS EQUAL MAX NUMBER
*DCALL(UPDA GRAD)
$
!MINORITER = MINORITER + 1
*GOTO DOOPT
$
$   EXACT CONSTRAINT GRADIENTS
$
*LABEL NOUP
!NNST = 1           $ NOTE: NLST = NUMBER OF LOAD SETS
*LABEL BLADSET
*IF("NNST" GT "NLST"): *GO TO ELADSET        $ LOOP OVER LOAD SETS
  !LCAS=DS,1,"NNST",1(1,LSET,CASE,MASK,MASK)  $ number of cases in set
  !OLIB = 1
```

```
* IF("GLIM" LT 0.0):*DCALL(SENS FILT)
* DCALL(CALC SENS)      $ CALCULATE SENSITIVITIES
  !STATUS = 2
* DCALL(DISP CONS)      $ ADD DISPLACEMENT CONSTRAINTS TO DATA SETS
  !NNST = NNST + 1
*GO TO BLADSET
*LABEL ELADSET
!STATUS = 2
* DCALL(ADS CONS)      $ WRITE OUT EXACT CONSTRAINT GRADIENTS IN ADS FORMAT
!MAJORITER = MAJORITER + 1
!MINORITER = 0
!EXACT = 1
$
*LABEL DOOPT
*XQT AUS
 DEFINE X = 1 X ADS 1 1
 TABLE,U(TYPE=-1): "SLIB" X ADS 1 1
  TRANSFER(SOURCE=X,OPER=XSUM)
 DEFINE F = 1 F ADS 1 1
 TABLE,U(TYPE=-1): "SLIB" F ADS 1 1
  TRANSFER(SOURCE=F,OPER=XSUM)
 DEFINE G = 1 G ADS 1 1
 TABLE,U(TYPE=-1): "SLIB" G ADS 1 1
  TRANSFER(SOURCE=G,OPER=XSUM)
!OLD = DS,1,1,1(1,F,ADS,1,1)
$ ML stuff
!ICT = ICT + 1
$
*XQT DCU
*IF("FIRST" NE 0): COPY 1 3 COMP DV 1 1  $COPY DV'S TO LIB 3 TO SAVE CYCLE DATA
$
*DCALL(CALC ML)  $ calculate move limits for this LP execution
*TF STANDARD OUTPUT=19  $redirect optimizer output
!MAJORITER
!MINORITER
!NEWPEN
!ML
*XQT LIP4 $ CALL LINEAR PROGRAMMING PROC.
 RESET LPEN = 1 KPEN="KPEN"  $CUTO = 0.01
*XQT DCU $ TO GET OUT OF LIP
*TF STANDARD OUTPUT=6 $redirect output back to eal file
!OLDPEN=NEWPEN
*XQT AUS
 OUTLIB = "ALIB"
 INLIB = "ALIB"
 DEFINE GLIN = 1 G ADS 1 1
 TABLE,U(TYPE=-1): "SLIB" GLIN ADS 1 1
  TRANSFER(SOURCE= GLIN, OPER= XSUM)
!FIRST = 0  $ flag indicates at least one cycle is complete
*XQT AUS
 OUTLIB = "ALIB":  INLIB = "ALIB"
 DEFINE G = 1 G ADS 1 1
 GMAX = MM1(G) $ find max constraint value
 !GMAX = DS,1,1,1("ALIB" GMAX AUS MASK MASK)$$$$
!GMAX
$
*XQT DCU
 COPY 1 3 COMP DV 1 1  $COPY DESIGN VARIABLES TO LIBRARY 3 TO SAVE CYCLE DATA
```

```
$
*IF("ML" LT 0.005): *GOTO EMAJORITER $ ML TOO SMALL TO CONTINUE
!MMMM = MMAJOR * MMINOR + MMAJOR
*IF("ICT" GT "MMMM"): *GOTO EMAJORITER
*IF("MAJORITER" GE "MMAJOR"): *GO TO EMAJORITER   $ ITERATION LIMIT
*GO TO BMAJORITER
*LABEL EMAJORITER
$
*DCALL(CLEA NL01)      $ CLEAR UP LIBRARIES
!NNST = 1            $ NOTE: NLST = NUMBER OF LOAD SETS
*LABEL BSET
  !LCAS=DS,1,"NNST",1(1,LSET,CASE,MASK,MASK)
  !OLIB = 1
*  DCALL (CALC STAT)  $ STATIC ANALYSIS OF FINAL DESIGN
*  XQT CONS
    RESET RHOK = "RHOKS" SET="NNST" NOUT="OLIB" BLEN="BLEN"  LLIB="LLIB"
    RESET NOJUNK=YES
  !STATUS=1
*  DCALL (DISP CONS)
  !NNST = NNST + 1
*IF("NNST" GT "NLST"): *GO TO ESET        $ LOOP OVER LOAD SETS
*GO TO BSET
*LABEL ESET
!STATUS = 1
*DCALL(ADS CONS)       $ WRITE OUT CONSTRAINTS IN ADS FORMAT
$
*XQT DCU
 COPY 1 3 COMP DV 1 1  $COPY DESIGN VARIABLES TO LIBRARY 3 TO SAVE CYCLE DATA
$
$ SAVE REGISTERS HERE
*XQT U1
*  REGI STORE(1 REGI DRIV 1 1)
*ECHO 4
$ EXITING: (DRIV OPT)
$ DRIVER PROGRAM TO OPTIMIZE A STRUCTURE SUBJECT TO STRESS AND DISP CONSTRAINTS
*NOECHO 4
$
*RETURN
*              ENDOPT
$
$-------------------------------------------------------------------------------*
$
*(CALC ML)   ENDML
*ECHO 4
$ ENTERING: (CALC ML)
$ CALCULATE MOVE LIMITS
$ ASSUMES THAT X IS POSITIVE
*NOECHO 4
$
$ GET DESIGN VARIABLES FROM X ADS, LIMITS FROM COMP DV
$ INCORPORATES MINIMUM ML RANGE BASED ON MIN GAUGE TOO
*XQT AUS
 INLIB= "ALIB"
 OUTLIB= "ALIB"
 DEFINE X = 1 X ADS 1 1
*IF("ML" LT 0.0): *GOTO SKIP
  DEL = UNION ("ML" X)
  TABLE,U(TYPE=-1):"SLIB" DXML
```

```
   TRANSFER(SOURCE=DEL, OPER= XSUM)
*LABEL SKIP
 DEFINE DXML = "SLIB" DXML
 DE1 : SOURCE = 1 COMP DV 1 1 : JS = 1 DESV MAP MASK MASK
     IS = 4 : DEST= LB : EX1
     IS = 5 : DEST= UB : EX1
 MLMIN = UNION(0.1 LB) $ MAKE MIN ML TO BE 10% OF LOWER BOUND
 ML = MAX(DXML,MLMIN)
 LBML = SUM(X, -1.0 ML)
 UBML = SUM(X,  1.0 ML)
 TLB = MAX(LBML, LB)
 TUB = MIN(UBML, UB)
 TABLE,U(TYPE=-1): 1 XB ADS 1 1
  TRANSFER(SOURCE=TLB, OPER=XSUM, ILIM=1, JLIM= "NDVA", DSKIP=1)
  TRANSFER(SOURCE=TUB, OPER=XSUM, ILIM=1, JLIM= "NDVA", DSKIP=1, DBASE=1)
$
*ECHO 4
$ EXITING: (CALC ML) $ CALCULATE ML
*NOECHO 4
*RETURN
*             ENDML
$
$-------------------------------------------------------------------------*
$
*(ADS INIT)       ENDADS
$*ECHO 1 2 3 4
*ECHO 4
$ ENTERING: (ADS INIT)
$ INITIALIZE ADS DATA
*NOECHO 4
$
$ ADS INIT CREATES TABLES NEEDED BY THE OPTIMIZER - MOSTLY JUST SPACE
$ IS RESERVED BUT SOME INFO IS TAKEN FROM COMP DV.
$ DATASETS STILL HAVE ADS IN THEIR NAMES FROM THE OLD DAYS BUT ARE
$ CURRENTLY IN MINOS READABLE FORMAT.
!NDVA=0
!NDVB=0
!NCON=0
!NDVI = 0
*LABEL BDVALOOP
!NDVI = NDVI +1
*IF("NDVI" GT "NDV"): *GO TO EDVALOOP   $LOOP THROUGH DESIGN VARIABLES
 !STA=DS,7,'NDVI",1(1,COMP,DV,MASK,MASK)  $ GET COEFFICIENT OF A+B*DV
 !STA=STA+.99999
 !STT=IFIX(STA)  $IF B IS NOT ZERO DESIGN VARIABLE IS ACTIVE AND UPDATE COUNTER
* IF("STT" NE 0): !NDVA=NDVA+1  $  NDVA IS NUMBER OF ACTIVE DESIGN VARIABLES
*GO TO BDVALOOP
*LABEL EDVALOOP
*XQT AUS
 TABLE(NI=1,NJ="NDVA",TYPE=-1): X ADS $INITIALIZE TABLE FOR D.V'S FOR ads
 TABLE(NI=1,NJ="NDVA",TYPE=-1): "SLIB" X ADS $INITIALIZE TABLE FOR D.V'S
 TABLE(NI="NLST",NJ="NDVA",TYPE=-1): DR X : INIT= "DR" $ INITIAL DR'S
 TABLE(NI=1,NJ="NDVA",TYPE=-1): "SLIB" DXML $INITIALIZE TABLE FOR D.V. ML
 TABLE(NI="NLST",NJ="NDVA",TYPE=-1): DR DIF  $ INITIAL TABLE FOR FD ERRORS
 TABLE(NI=2,NJ="NDVA",TYPE=-1): XB ADS  $INITIALIZE TABLE FOR BOUNDS FOR ADS
 TABLE(NI=1,NJ="NDVA",TYPE=0):DESV MAP  $INITIALIZE TABLE FOR MAPPING ACTIVE DV'S TO DV LIST
 TABLE(NI=1,NJ=1,TYPE=-1): F ADS 1 1 $ INITIALIZE TABLE FOR OBJ
 TABLE(NI=1,NJ=1,TYPE=-1): "SLIB" F ADS 1 1 $ INITIALIZE TABLE FOR OBJ
```

```
 TABLE(NI="NDVA",NJ=1,TYPE=-1) : DF ADS 1 1 $ INITIALIZE TABLE FOR OBJ GRADIENT
 TABLE(NI="NDVA",NJ=1,TYPE=-1) : "SLIB" DF ADS 1 1 $ INITIALIZE TABLE FOR OBJ GRADIENT
 !STATUS = 0
 TABLE(NI=5,NJ=1,TYPE=0): STAT ADS $ ADS OPTIMIZATION OPTIONS
  J=1: "STATUS" 8 4 3 1
 !NDVI = 0
* LABEL BDVBLOOP
 !NDVI = NDVI +1
* IF("NDVI" GT "NDV"): *GO TO EDVBLOOP   $LOOP THROUGH DESIGN VARIABLES
   !STA=DS,7,"NDVI",1(1,COMP,DV,MASK,MASK)  $ GET COEFFICIENT OF A+B*DV
   !STA=STA+.99999
   !STT=IFIX(STA) $IF B IS NOT ZERO DESIGN VARIABLE IS ACTIVE AND UPDATE COUNTER
*  IF("STT" EQ 0): *GO TO BDVBLOOP
    !NDVB=NDVB+1  $ ACTIVE D.V. NUMBER
    !SECT=DS,1,"NDVI",1(1,COMP,DV,MASK,MASK)
    !V=DS,3,"NDVI",1(1,COMP,DV,MASK,MASK)      $GET DESIGN VARIABLE
    !UB=DS,5,"NDVI",1(1,COMP,DV,MASK,MASK)     $GET LOWER BOUND
    !LB=DS,4,"NDVI",1(1,COMP,DV,MASK,MASK)     $GET UPPER BOUND
    TABLE,U (TYPE=0):DESV MAP: OPER=XSUM: J="NDVB":"NDVI" $WRITE MAPPING TO TABLE
    TABLE,U (TYPE=-1):X ADS:  OPER=XSUM: J="NDVB":"V"   $WRITE D.V TO TABLE
    TABLE,U (TYPE=-1):XB ADS: OPER=XSUM:I=1 2: J="NDVB":"LB","UB"  $WRITE L. B. TO TABLE
* GO TO BDVBLOOP
* LABEL EDVBLOOP
$
$ determine total number of groups for each element type
!GE21=TOC,NJ(1 GD E21 MASK MASK)  $ NUMBER OF E21 GROUPS
!GE23=TOC,NJ(1 GD E23 MASK MASK)  $ NUMBER OF E23 GROUPS
!GE33=TOC,NJ(1 GD E33 MASK MASK)  $ NUMBER OF E33 GROUPS
!GE43=TOC,NJ(1 GD E43 MASK MASK)  $ NUMBER OF E43 GROUPS
!GE31=TOC,NJ(1 GD E31 MASK MASK)  $ NUMBER OF E31 GROUPS
!GE32=TOC,NJ(1 GD E32 MASK MASK)  $ NUMBER OF E32 GROUPS
!GE41=TOC,NJ(1 GD E41 MASK MASK)  $ NUMBER OF E41 GROUPS
!GE42=TOC,NJ(1 GD E42 MASK MASK)  $ NUMBER OF E42 GROUPS
!GE44=TOC,NJ(1 GD E44 MASK MASK)  $ NUMBER OF E44 GROUPS
$ Create tables of active grps for each element type
$ by examining comp grp table - added 8/9/94
!NACT=TOC,NJ (1 COMP GRP MASK MASK)
* IF("GE21" GT 0 ): TABLE(NI=1,NJ="GE21",TYPE=0):GACT E21
* IF("GE23" GT 0 ): TABLE(NI=1,NJ="GE23",TYPE=0):GACT E23
* IF("GE33" GT 0 ): TABLE(NI=1,NJ="GE33",TYPE=0):GACT E33
* IF("GE43" GT 0 ): TABLE(NI=1,NJ="GE43",TYPE=0):GACT E43
* IF("GE31" GT 0 ): TABLE(NI=1,NJ="GE31",TYPE=0):GACT E31
* IF("GE32" GT 0 ): TABLE(NI=1,NJ="GE32",TYPE=0):GACT E32
* IF("GE41" GT 0 ): TABLE(NI=1,NJ="GE41",TYPE=0):GACT E41
* IF("GE42" GT 0 ): TABLE(NI=1,NJ="GE42",TYPE=0):GACT E42
* IF("GE44" GT 0 ): TABLE(NI=1,NJ="GE44",TYPE=0):GACT E44
!LOOP=0
*LABEL GLOOP
 !LOOP=LOOP+1
*IF("LOOP" GT "NACT"): *GO TO EGLOOP
 !EXX=DS,3,"LOOP",1 (1,COMP GRP 1 1)
 !NGRP=DS,4,"LOOP",1 (1,COMP GRP 1 1)
 TABLE,U(TYPE=0): GACT "EXX"
   I=1: J="NGRP": 1
*GO TO GLOOP
*LABEL EGLOOP
*IF("GLIM" GE 0.0): *GO TO EGCOPY
*XQT DCU
```

```
  COPY 1 "SLIB" GACT E21
  COPY 1 "SLIB" GACT E23
  COPY 1 "SLIB" GACT E31
  COPY 1 "SLIB" GACT E32
  COPY 1 "SLIB" GACT E33
  COPY 1 "SLIB" GACT E41
  COPY 1 "SLIB" GACT E42
  COPY 1 "SLIB" GACT E43
  COPY 1 "SLIB" GACT E44
*XQT AUS
*LABEL EGCOPY
$compute number of displacement constraints in each loadset
!LOOP=0
*LABEL LSLOOP
 !LOOP=LOOP+1
*IF("LOOP" GT "NLST"): *GO TO ELSLOOP
 !NCAS=DS,1,"LOOP",1 (1,LSET CASE 1 1)
 !DISC=NUMD*NCAS
 TABLE,U(TYPE=0): LSET CASE: OPER=XSUM
  I=2: J="LOOP": "DISC"
*GO TO LSLOOP
*LABEL ELSLOOP
!LOOP=FREE()
!EXX=FREE()
!NGRP=FREE()
!NACT=FREE()
$
*XQT DCU
!NNST=1
*LABEL LOOPER
$RENAME ALL DATA SETS AS INDICATED BELOW
 COPY 1 "SLIB" APPL FORC "NNST" 1
 CHANGE 1 APPL FORC "NNST" 1,ORIG FORC "NNST" 1
 COPY "SLIB" 1 APPL FORC "NNST" 1
!NNST=NNST+1
*IF("NLST" GE "NNST") : *GO TO LOOPER
$Clean up registers
!V=FREE()
!UB=FREE()
!LB=FREE()
!SECT=FREE()
!NDVB=FREE()
!NDVI=FREE()
!STT=FREE()
!STA=FREE()
$
*ECHO 4
$ EXITING: (ADS INIT) $ INITIALIZE ADS DATA
*NOECHO 4
*RETURN
*             ENDADS
$
$-------------------------------------------------------------------------*
$
*(CALC STAT)   ENDSTA
*ECHO 4
$ ENTERING: (CALC STAT)
$ PERFORM STATIC ANALYSES OF STRUCTURE
```

```
*NOECHO 4
$
$ CALC STAT INCORPORATES THE DESIGN VARIABLES AND FINITE ELEMENT MODEL
$ AND CONTROLS THE EXECUTION OF A STATIC ANALYSIS
$ GET DESIGN VARIABLES FROM X ADS, LIMITS FROM COMP DV
$
$!ML
*XQT AUS
 !NDVI = 1
* LABEL BDVLOOP
* IF("NDVI" GT "NDVA"): *GO TO EDVLOOP      $ LOOP THROUGH DESIGN VARIABLES
   !STT=DS,1,"NDVI",1(1 DESV MAP MASK MASK) $ GET MAPPING OF ACTIVE DV
   !V = DS,1,"NDVI",1(1 X ADS 1 1)        $ GET DESIGN VARIABLE FROM ADS
   TABLE,U (TYPE=-1):COMP DV 1 1: OPER=XSUM
    I=3: J="STT": "V"   $ RESET DESIGN VARIABLE
    I=8: J="STT": 0.0   $ RESET PROP EVALUATION FLAG
 !NDVI = NDVI +1
* GO TO BDVLOOP
* LABEL EDVLOOP
*XQT DCU $ JUST TO GET THE ABOVE STUFF TO COMPLETELY EXECUTE
!ISEN=0
*DCALL(SECT INIT)$ WRITE  D.V. DEFINITIONS TO ELEMENT EFILE
$ ASSEMBLE STIFFNESS MATRIX USING EAL PROCESSORS
*XQT EKS
*XQT K
*XQT RSI
 RESET  CON="NNST"
*XQT EQNF
 RESET SET="NNST"  L2="LCAS"
*XQT DCU
$RENAME ALL DATA SETS AS INDICATED BELOW
 COPY 1 "SLIB" ORIG FORC "NNST" 1
 CHANGE "SLIB" ORIG FORC "NNST" 1,APPL FORC "NNST" 1
 COPY "SLIB" 1 APPL FORC "NNST" 1
 PACK 9
*XQT AUS
!ZCAS=1
*LABEL ZLCASE
$ IF EQNF FORC DATA SET EXISTS - ADD TO APPL FORC
!EQFO=TOC, RR (1 EQNF FORC "NNST" "ZCAS")
*IF("EQFO" EQ 0 ): *GO TO ZEND
 DEFI EQIL = 1 EQNF FORC "NNST" "ZCAS"
 TABLE,U(TYPE=-1): APPL FORC "NNST" 1
 TRAN(SOUR=EQIL,OPER=SUM,L1="ZCAS",L2="ZCAS")
*LABEL ZEND
!ZCAS=ZCAS+1
*IF("LCAS" GE "ZCAS"): *GO TO ZLCASE
*XQT DCU
!ZCAS=1
*LABEL CAS2
 DISABLE 1 EQNF FORC "NNST" "ZCAS"
!ZCAS=ZCAS+1
*IF("LCAS" GE "ZCAS"): *GO TO CAS2
*XQT SSOL  $CALCULATE DISPLACEMENTS FOR LOADSET NNST
 RESET SET="NNST" CON="NNST"
!NN3=TOC,N3(1,K,SPAR,MASK,MASK) $ GET FULL K SPAR DATASET NAME
!NN4=TOC,N4(1,K,SPAR,MASK,MASK)
!NAM1='STAT
```

```
!NAM2=DISP
!NAM3=NNST
!NAM4=NNST
$ OLIB IS WHERE STRESS RESULTANTS FROM *XQT ES GO
*DCALL(CALC N)  $CALCULATE STRESS RESULTANTS USING ES PROCESSOR
!NAM1=FREE()
!NAM2=FREE()
!NAM3=FREE()
!NAM4=FREE()
*XQT DCU
$RENAME ALL DATA SETS AS INDICATED BELOW
 COPY 1 "SLIB" STAT DISP "NNST" "NNST"
 CHANGE 1 STAT DISP "NNST" "NNST",PREV DISP "NNST" "NNST"
 COPY "SLIB" 1 STAT DISP "NNST" "NNST"
 COPY 1 "SLIB" STAT REAC "NNST" "NNST"
 CHANGE 1 STAT REAC "NNST" "NNST",PREV REAC "NNST" "NNST"
 COPY "SLIB" 1 STAT REAC "NNST" "NNST"
 CHANGE 1 APPL FORC "NNST" 1,PREV FORC "NNST" 1
 DISABLE 1 DEM DIAG 0 0
 CHANGE 1 K SPAR "NN3" "NN4" KP SPAR "NN3" "NN4"
 CHANGE 1 INVA K "NNST" 0,INVA KP "NNST" 0
 CHANGE 1 XINV K "NNST" 0,XINV KP "NNST" 0
 PACK 9
*IF("NNST" GT 1): *GO TO ENDOFSTAT $ DO THE FOLLOWING ONLY FOR FIRST LOAD SET
$ CALCULATE TOTAL STRUCTURAL WEIGHT - OBJECTIVE FUNCTION
$ THIS IS DONE BECAUSE EAL PROCESSOR E IS NOT EXECUTED FOR SENSITIVITIES
* XQT AUS
*  DCALL(CALC MASS) $ PROCEDURE TO COMPUTE TOTAL MASS
  TABLE,U(TYPE=-1): 1 F ADS 1 1
   OPER=XSUM: I=1: J=1: "MASS"
* XQT DCU $ TO GET OUT OF AUS
! OMAS=MASS
*  ECHO 4
$   MASS OF THE STRUCTURE IS:
*  NOECHO 4
  !MASS
*LABEL ENDOFSTAT
*ECHO 4
$ EXITING: (CALC STAT) $ PERFORM STATIC ANALYSES OF INITIAL (UNPERTURBED) STRUCTURE
*NOECHO 4
$
*RETURN
*              ENDSTA
$
$-------------------------------------------------------------------------*
$
*(CALC N)         ENDCN
*ECHO 4
$ ENTERING: (CALC N)
$ CALCULATE STRESS RESULTANTS
*NOECHO 4
$ THIS PROCEDURE USES ES TO CALCULATE THE STRESS RESULTANTS
$ FOR THE ACTIVE DESIGN REGIONS
$
*XQT ES
 PMODE=2                    $ INHIBIT PRINTOUT
 OUTLIB= "OLIB"
 U= 1 "NAM1" "NAM2" "NAM3" "NAM4"   $ CURRENT DISPLACEMENTS
```

```
 NODES=0                    $ ELEMENT CENTER ONLY
 OUTPUT TYPE = SR
* IF("GE21" EQ 0): *GO TO 2500
  !IE=GE21
  !NAM5='E21
*  DCALL(STRS GRP) $ CALCULATE E21 STRESS RESULTANTS
* LABEL 2500
* IF("GE23" EQ 0): *GO TO 2501
  !IE=GE23
  !NAM5='E23
  OUTPUT TYPE=ES
*  DCALL(STRS GRP) $ CALCULATE E23 STRESS RESULTANTS
  OUTPUT TYPE = SR
* LABEL 2501
* IF("GE31" EQ 0): *GO TO 2502
  !IE=GE31
  !NAM5='E31
*  DCALL(STRS GRP) $ CALCULATE E31 STRESS RESULTANTS
* LABEL 2502
* IF("GE32" EQ 0): *GO TO 2503
  !IE=GE32
  !NAM5='E32
*  DCALL(STRS GRP) $ CALCULATE E32 STRESS RESULTANTS
* LABEL 2503
* IF("GE33" EQ 0): *GO TO 2504
  !IE=GE33
  !NAM5='E33
*  DCALL(STRS GRP) $ CALCULATE E33 STRESS RESULTANTS
* LABEL 2504
* IF("GE41" EQ 0): *GO TO 2505
  !IE=GE41
  !NAM5='E41
*  DCALL(STRS GRP) $ CALCULATE E41 STRESS RESULTANTS
* LABEL 2505
* IF("GE42" EQ 0): *GO TO 2506
  !IE=GE42
  !NAM5='E42
*  DCALL(STRS GRP) $ CALCULATE E42 STRESS RESULTANTS
* LABEL 2506
* IF("GE43" EQ 0): *GO TO 2507
  !IE=GE43
  !NAM5='E43
*  DCALL(STRS GRP) $ CALCULATE E43 STRESS RESULTANTS
* LABEL 2507
* IF("GE44" EQ 0): *GO TO 2508
  !IE=GE44
  !NAM5='E44
  OUTPUT TYPE=ES
*  DCALL(STRS GRP) $ CALCULATE E44 STRESS RESULTANTS
* LABEL 2508
!IE=FREE()
!NAM5=FREE()
*ECHO 4
$ EXITING: (CALC N) $ CALCULATE STRESS RESULTANTS
*NOECHO 4
$
*RETURN
*           ENDCN
```

```
$
$-------------------------------------------------------------------------*
$
*(CALC SENS)      ENDSEN
*ECHO 4
$ ENTERING: (CALC SENS)
$ CALCULATE DK/DV, DM/DV, STRESS RESULTANT AND DISPLACEMENT DERIVATIVES
*NOECHO 4
$
$ this routine loops thru all the active design variables, perturbs them, and
$ controls the calculation of the sensitivities
$
!OLIB = OTIB
!IIV=1
!ISEN = 1
*LABEL BDVLOOP  $ loop thru active design variables
*IF("IIV" GT "NDVA"): *GO TO EDVLOOP
 !IIVV=DS,1,"IIV",1(1,DESV,MAP,MASK,MASK)
 !DVP=DS,3,"IIVV",1(1,COMP,DV,MASK,MASK)  $ GET DESIGN VARIABLE
 !BQ = ABS(DVP)
 !DR = DS,"NNST","IIV",1(1,DR,X,MASK,MASK) $ GET DR FOR DESIGN VARIABLE
* IF("BQ" GT "DR") : !DV=DR*DVP+DVP : !DELV=DR*DVP  $NORMAL PERTURBATION
* IF("BQ" LE "DR") : !DELV = DR*DR  : !DV=DVP + DELV $PERTURBATION FOR SMALL DVP
 !AQ=1./DELV
 !BQ=-1.*AQ
* XQT AUS
 !NBLK = TOC,NBLOCKS(1 STRS DCON "NNST" "IIV")
* IF("NBLK" GT 0): *GO TO EXISTS
$  determine size of constraint dataset and create
  !NINJ = TOC,NINJ(1 STRS CONS "NNST" 1)
  !NJ  = TOC,NJ  (1 STRS CONS "NNST" 1)
  !NI=NINJ/NJ
  TABLE(NI="NI",NJ="NJ",TYPE=-1): 1 STRS DCON "NNST" "IIV"
* LABEL EXISTS
  TABLE,U(TYPE=-1):COMP DV$ UPDATE DESIGN VARIABLE LIST
   OPER=XSUM
   I=3 : J="IIVV":"DV"  $Write PERTURBED DESIGN VARIABLE TO LIST
   I=8 : J="IIVV":-1.   $SET FLAG TO INDICATE DESIGN VARIABLE IS PERTURBED
* XQT DCU
 ERASE "OLIB"
 DISABLE 1 APPL FORC 50  MASK
 DISABLE 1 STAT DISP 50  MASK
 DISABLE 1 STAT REAC 50  MASK
 !MARK=0
*  LABEL BSRDELETE  $ disable all ES Exx datasets
  !DUMM = TOC,NWDS(1 SR  MASK MASK MASK) MARK
*  IF("MARK" EQ -1): *GO TO ESRDELETE
  DISABLE 1 "MARK"
*  GOTO BSRDELETE
*  LABEL ESRDELETE
 !MARK=0
*  LABEL BESDELETE $ disable all ES Exx datasets
  !DUMM = TOC,NWDS(1 ES  MASK MASK MASK) MARK
*  IF("MARK" EQ -1): *GO TO EESDELETE
  DISABLE 1 "MARK"
*  GOTO BESDELETE
*  LABEL EESDELETE
 PACK 1 $ clean out all disabled datasets from LIB 1
```

```
* DCALL (SECT INIT)  $WRITE NEW STIFFNESSES FOR PERTURBED VARIABLE
* XQT AUS
  INLIB ="ALIB"
  OUTLIB ="ALIB"
*  IF("NNST" GT 1): *GO TO ENDOFDWT $ DO THE FOLLOWING ONLY FOR FIRST LOAD SET
$   CALCULATE GRADIENT OF TOTAL STRUCTURAL WEIGHT - OBJECTIVE FUNCTION
* DCALL(CALC MASS)  $ calc mass of perturbed structure
! DELM=MASS-OMAS   $ change in mass
! GOBJ=AQ*DELM     $ weight gradient
    TABLE,U(TYPE=-1):1 DF ADS 1 1  $TABLE OF DERIVATIVES OF OBJECTIVE FUNCTION
     OPER=XSUM: I="IIV":J=1:"GOBJ"
    TABLE,U(TYPE=-1):"SLIB" DF ADS 1 1  $ TABLE OF DERIVATIVES OF OBJECTIVE FUNCTION
     OPER=XSUM: I="IIV":J=1:"GOBJ"
*  ECHO 4
$    WEIGHT GRADIENTS
  !IIV
  !GOBJ
*  NOECHO 4
  !GOBJ=FREE()
  !DELM=FREE()
* LABEL ENDOFDWT
* DCALL(CALC DUDV)     $ CALCULATE DISPLACEMENT DERIVATIVES
* DCALL(CALC DELN)     $ CALCULATE STRESS RESULTANT PERTURBATIONS
* XQT DCU            $ PUT PERTURBED DISPLACEMENTS IN OLIB
  COPY 1 "OLIB" U DELU "NNST" 1
  CHANGE "OLIB" U DELU "NNST" 1 STAT DISP "NNST" "NNST"
  COPY 1 "OLIB" R DELU "NNST" 1
  CHANGE "OLIB" R DELU "NNST" 1 STAT REAC "NNST" "NNST"
 ERASE "ALIB"
 PRIN 1 COMP DV
* XQT AUS
* XQT CONS    $ PROCESSOR TO EVALUATE CONSTRAINTS (stress & local buckling)
  RESET RHOK = "RHOKS" SET="NNST" NOUT="OLIB" BLEN="BLEN"  LLIB="LLIB"
  RESET NOJUNK=  YES
 !STATUS = 1
* DCALL(DISP CONS) $ compute displacement constraints
*XQT AUS
$ COMPUTE CONSTRAINT SENSITIVITIES BY FINITE DIFFERENCE
 DEFINE OCON = 1 STRS CONS "NNST" 1
 DEFINE NCON = "OLIB" STRS CONS "NNST" 1
 TABLE,U(TYPE=-1): 1 STRS DCON "NNST" "IIV" $ SUM("AQ" NCON, "BQ" OCON)
  TRANSFER(SOURCE=OCON,OPER=XSUM,C="BQ")     $ ,L2="LCAS")
  TRANSFER(SOURCE=NCON,OPER=SUM ,C="AQ")     $ ,L2="LCAS")
$
TABLE,U(TYPE=-1):COMP DV   $ RESET DESIGN VARIABLE TO ORIGINAL VALUE
 OPER=XSUM: I=3 : J="IIVV":"DVP"
*  XQT MBED         $ RESET SECTION DATA TO UNPERTURBED STATE
   RESET COMP="JS" NOUT="ALIB"
*XQT DCU
*DCALL(ELEM MBED)  $return e-state to unperturbed condition
* XQT EKS
 SELECTION="ALIB" ETAB
!IIV = IIV + 1
*GO TO BDVLOOP
*LABEL EDVLOOP
*ECHO 4
$ EXITING: (CALC SENS) $ CALCULATE DK/DV, DM/DV, STRESS CONSTRAINT AND DISPLACEMENT
DERIVATIVES
```

```
*NOECHO 4
$
*RETURN
*             ENDSEN
$
$-------------------------------------------------------------------------*
$
*(CALC DUDV)      ENDDUV    $CALCULATE DU/DCAPV - DISPLACEMENT DERIVATIVES
*ECHO 4
$ ENTERING: (CALC DUDV)
$ CALCULATE DISPLACEMENT DERIVATIVES
*NOECHO 4
$ THIS ROUTINE CALCULATES THE DISPLACEMENT DERIVATIVES BY SEMI-ANALYTIC
$ METHOD - DK/DV COMPUTED BY FINITE DIFFERNCE. ONLY PORTION OF K MATRIX
$ (LSK LS DATASETS) AFFECTED BY DV CHANGE IS DIFFERENCED
$
!LL3=TOC,N3("OLIB",LSK,LS,MASK,MASK) $ GET FULL LSK LS DATASET NAME
!LL4=TOC,N4("OLIB",LSK,LS,MASK,MASK)
*XQT AUS
   DEFINE KN= "OLIB" LSK LS "LL3" "LL4"  $ NEW K MATRIC SECTION
   DEFINE KO= "ALIB" LSK LS "LL3" "LL4"  $ OLD K MATRIC SECTION
   DEFINE U=PREV DISP "NNST" "NNST"     $ UNPERTURBED DISPLACEMENT VECTOR
 "SLIB" KU VECT "NNST" = PROD(KO,U)
 "OLIB" KU VECT "NNST" = PROD(KN,U)
  DEFINE KUH = "OLIB" KU VECT "NNST"
  DEFINE KUL = "SLIB" KU VECT "NNST"
   DKU =SUM("AQ" KUL, "BQ" KUH)   $ DKU = -dK/dV*u
$the following section calculates DF/DV (force sensitivities) if either
$an EQNF FORC or intertia force (grav.ne.0) is included
*XQT EQNF
 RESET SET="NNST"  L2="LCAS"
*XQT DCU
$RENAME ALL DATA SETS AS INDICATED BELOW
  COPY 1 "SLIB" ORIG FORC "NNST" 1
  CHANGE "SLIB" ORIG FORC "NNST" 1,APPL FORC "NNST" 1
  COPY "SLIB" 1 APPL FORC "NNST" 1
  PACK 9
*XQT AUS
!ZCAS=1
*LABEL ZLCASE
!EQFO=TOC, RR (1 EQNF FORC "NNST" "ZCAS")
*IF("EQFO" EQ 0 ): *GO TO ZEND
 DEFI EQIL = 1 EQNF FORC "NNST" "ZCAS"
 TABLE,U(TYPE=-1): APPL FORC "NNST" 1
 TRAN(SOUR=EQIL,OPER=SUM,L1="ZCAS",L2="ZCAS")
*LABEL ZEND
!ZCAS=ZCAS+1
*IF("LCAS" GE "ZCAS"): *GO TO ZLCASE
$FINITE DIFFERENCE OF LOAD VECTOR
 DEFI NEWF=APPL FORC "NNST" 1
 DEFI OLDF=PREV FORC "NNST" 1
 DELF=SUM("AQ" NEWF, "BQ" OLDF)
 APPL FORC 50 = SUM(DELF,DKU) ${dF/dV-dK/dV*u}
*XQT DCU
 DISABLE 1 APPL FORC "NNST" MASK
!ZCAS=1
*LABEL CAS2
 DISABLE 1 EQNF FORC "NNST" "ZCAS"
```

```
!ZCAS=ZCAS+1
*IF("LCAS" GE "ZCAS"): *GO TO CAS2
$ end of DF/DV calculation
$
$ clean up old datasets
 DISABLE 1 M AUS MASK MASK
 DISABLE 1 WT FORC "NNST" 1
 DISABLE 1 DELF AUS
 DISABLE 1 DKU  AUS
 PACK 1
*XQT SSOL $ static solution [Kinv]*{dF/dV-dK/dV*u}
 RESET SET=50, CON ="NNST",  K=KP
*XQT AUS
!ICAS = 1
*LABEL BCASELOOP $ loop over load cases
*IF("ICAS" GT "LCAS"): *GO TO ECASELOOP
*  IF("IIV" GT 1): *GO TO NOTFIRST
    !NBLK = TOC,NBLOCKS(1 DUDV CAPV "NNST" "ICAS")
$    SET UP DUDV CAPV DATASET IF REQUIRED (1st time thru)
*    IF("NBLK" GT 0): *GO TO NOTFIRST
     !NINJ = TOC,NINJ(1 STAT DISP 50 "NNST")
     !NJ  = TOC,NJ  (1 STAT DISP 50 "NNST")
     !NI=NINJ/NJ
     TABLE(NI="NI",NJ="NJ",TYPE=-1): DUDV CAPV "NNST" "ICAS"
      !IBLK=1
*      LABEL DULOOP  $ INITIALIZE ALL BLOCKS OF THE DATA SET
*      IF("IBLK" GT "NDVA"):*GO TO NOTFIRST
       BLOCK "IBLK"
       !IBLK = IBLK +1
*      GO TO DULOOP
*  LABEL NOTFIRST
  DEFI DU50=STAT DISP 50 "NNST" "ICAS", "ICAS"    $ DERIVATIVE OF DISPLACEMENT
  TABLE,U(TYPE=-1): DUDV CAPV "NNST" "ICAS"  $Put dU/dV in blk iiv of DUDV CAPV
   TRANSFER(SOURCE=DU50,OPERATION=XSUM,L1="IIV",L2="IIV")
!  ICAS = ICAS + 1
*GO TO BCASELOOP
*LABEL ECASELOOP
*ECHO 4
$ EXITING: (CALC DUDV) $ CALCULATE DISPLACEMENT DERIVATIVES
*NOECHO 4
$
*RETURN
*              ENDDUDV
$
$-------------------------------------------------------------------------*
$
*(CALC DELN)      ENDDELN
*ECHO 4
$ ENTERING: (CALC DELN)
$ CALCULATE PERTURBED DISPLACEMENTS AND STRESS RESULTANTS
*NOECHO 4
$  CALCULATE NEW DISPLACEMENTS BASED ON GRADIENT PREVIOUSLY CALCULATED
$  FOR EACH LOAD CASE - U(NEW)=U(OLD)+(DU/DV)*DELV
$  ROUTINE CALC N CALLED TO COMPUTE STRESS RESULTANTS FOR U(NEW)
*XQT AUS
 INLIB = "ALIB"
 OUTLIB = "ALIB"
 !NCAS = 1  $  LOOP OVER LOAD CASES
```

```
* LABEL BLCASE
* IF("NCAS" GT "LCAS"): *GO TO ELCASE
*   IF("NCAS" GT 1): *GO TO NOTFIRSTLC
     !NBLK = TOC,NBLOCKS(1 U DELU "NNST" 1)
$    SET UP U DELU DATASET IF REQUIRED
*    IF("NBLK" GT 0): *GO TO NOTFIRSTLC
      !NINJ = TOC,NINJ(1 PREV DISP "NNST" "NNST")
      !NJ  = TOC,NJ  (1 PREV DISP "NNST" "NNST")
      !NI=NINJ/NJ
      TABLE(NI="NI",NJ="NJ",TYPE=-1): 1 U DELU "NNST" 1
       !IBLK=1
*       LABEL ULOOP  $ INITIALIZE ALL BLOCKS OF THE DATA SET
*       IF("IBLK" GT "LCAS"):*GO TO NOTFIRSTLC
        BLOCK "IBLK"
        !IBLK = IBLK +1
*       GO TO ULOOP
*   LABEL NOTFIRSTLC          $ APPEND U DELU DISPLACEMENTS TO DATASET
   DEFINE U1= 1 PREV DISP "NNST" "NNST"  "NCAS","NCAS"
   DEFINE U2= 1 DUDV CAPV "NNST" "NCAS"  "IIV","IIV"
$  TRAN used instead of AUS SUM because its faster
$  CHANGED FROM AUS/SUM(U1,"DELV" U2)
   TABLE,U(TYPE=-1) : 1 U DELU "NNST" 1
    TRANSFER(SOURCE=U1,OPERATION=XSUM,L1="NCAS",L2="NCAS")
    TRANSFER(SOURCE=U2,C="DELV",OPERATION=SUM,L1="NCAS",L2="NCAS")
 !NCAS = NCAS + 1
* GO TO BLCASE
* LABEL ELCASE
$ LEAVING AUS HERE
!NAM1='U
!NAM2='DELU
!NAM3="NNST"
!NAM4='MASK
*DCALL(CALC N)  $ CALCULATE STRESS RESULTANTS BASED ON NEW DISPLACEMENTS
*ECHO 4
$ EXITING: (CALC DELN) $ CALC. PERTURBED DISP AND STR RESULTANTS
*NOECHO 4
*RETURN
*             ENDDELN
$
$-------------------------------------------------------------------------*
$
*(SECT INIT)      ENDINI $ WRITE SECTION DATA
*ECHO 4
$ ENTERING: (SECT INIT)
$ CHANGE SECTION PROPERTIES
*NOECHO 4
$ this routine writes the section properties from the design variables in
$ COMP DV. New section props written for any DV with a status flag (I=8)
$ of zero or -1.
$
!LSEC=0
!NDVI = 0
*LABEL BDVLOOP   $ loop over design variables
!NDVI = NDVI +1
$ check status flag in COMP DV for each design variable
$ if flag is zero or -1 a new section property is needed
*IF("NDVI" GT "NDV"): *GO TO EDVLOOP
  !STA=DS,8,"NDVI",1(1,COMP,DV,MASK,MASK) $ GET STA FLAG OF DESIGN VARIABLE
```

```
  !STT=IFIX(STA)            $ TO DETERMINE IF THE STIFFNESS NEEDS UPDATING
*  IF("STT" GT 0): *GO TO BDVLOOP
  !SECT=DS,1,"NDVI",1(1,COMP,DV,MASK,MASK)  $ GET COMPONENT NO.
  !JS=IFIX(SECT+.001)     $ FIND INTEGER OF SECT
* XQT AUS
 TABLE,U(TYPE=-1):COMP DV    $ RESET DESIGN VARIABLE TO ORIGINAL VALUE
  OPER=XSUM: I=8 : J="NDVI": 1.  $ RESET FLAG TO ZERO
*  IF("JS" EQ "LSEC"): *GO TO BDVLOOP $ WAS LAST UPDATED, DON'T GENERATE NEW STIFFNESS
MATRICES
  !LSEC=JS
   ERASE "ALIB"
$  MBED processor determines elements affected and writes stiffnesses based on
$  the current design variable data in COMP DV
*  XQT MBED                $ UPDATE SECTION DATA
   RESET COMP="JS" NOUT="ALIB"
*XQT DCU
$ execute the following 2 lines if sensitivities are not required (base design)
*   IF("ISEN" NE 0): *GO TO DOSENS
*    DCALL(ELEM MBED) $ embed stiffnesses in efile
*    GO TO BDVLOOP
* LABEL DOSENS
* XQT EKS  $ generate element stiffnesses for modified elements
  SELECTION="ALIB" ETAB
* XQT LSK     $ form unperturbed submatrix in ALIB
   RESET DEST= "ALIB"
   SELECTION = "ALIB" ETAB
*DCALL(ELEM MBED) $embed perturbed stiffnesses in efile
* XQT EKS  $ generate element stiffnesses for modified elements
  SELECTION="ALIB" ETAB
* XQT LSK     $ form perturbed submatrix in OLIB
   RESET DEST= "OLIB"
   SELECTION = "ALIB" ETAB
*XQT DCU
*GO TO BDVLOOP
*LABEL EDVLOOP
!SECT=FREE()  $clean up old registers
!NDVI=FREE()
!STT=FREE()
!STA=FREE()
!LSEC=FREE()
*ECHO 4
$ EXITING: (SECT INIT) $ CHANGE SECTION PROPERTIES
*NOECHO 4
$
*RETURN
*              ENDINI
$
$-------------------------------------------------------------------------*
$
*(STRS GRP)       ENDGRP
$ loops over all groups of element type Exx="NAM5" and specifies stress
$ computation for all groups labeled as active in GACT "NAM5"
!IV = 0
*LABEL BGROUPLOOP
!IV = IV + 1
*IF("IV" GT "IE") : *GO TO EGROUPLOOP
!GACT = DS,1, "IV",1 (1, GACT "NAM5" 1 1)
*IF("GACT" EQ 0): *GO TO BGROUPLOOP  $ group not active
```

```
  "NAM5"  "IV"  $ CREATE ES TABLE FOR EXX GROUP IV
*GO TO BGROUPLOOP
*LABEL EGROUPLOOP
!IV = FREE()
*RETURN
*             ENDGRP
$
$-------------------------------------------------------------------------*
$
*(DISP CONS)      ENDCON
*ECHO 4
$ ENTERING: (DISP CONS)
$ DISPLACEMENT CONSTRAINTS AND DERIVATIVES
*NOECHO 4
$ routine gets displacement limits from DEFL DEFI dataset and
$ then gets calculated deflection from stat disp and formulates
$ the constraint or sensitivity value and writes to STRS CONS
$ or STRS DCON dataset
$
*XQT AUS
 !CNOD = 0
* LABEL BCNODELOOP  $ LOOP FOR CONSTRAINED NODES
 !CNOD = CNOD + 1
* IF("CNOD" GT "NUMD"): *GO TO ECNODELOOP
  !XNOD=DS,1,"CNOD",1(1,DEFL DEFI,MASK,MASK)$ NODE NO.
  !NOD=IFIX(XNOD)
  !DIR=DS,2,"CNOD",1(1,DEFL,DEFI,MASK,MASK) $ DIRECTION
  !NDIR=IFIX(DIR)
  !LIM=DS,3,"CNOD",1(1,DEFL,DEFI,MASK,MASK) $ LIMIT
  !NC = 1
*  LABEL BCASELOOP $ LOOP OVER CASE NUMBER
*  IF("NC" GT "LCAS"): *GO TO ECASELOOP
*   IF("STATUS" EQ 2): *GO TO DERIVS
$    CONSTRAINTS ONLY, "OLIB" ASSUMED FOR STAT DISP & STRS CONS
     !DEL=DS,"NDIR","NOD","NC"("OLIB",STAT,DISP,"NNST","NNST")
     !CON=DEL/LIM-1. $ DISPLACEMENT CONSTRAINT VALUE
     !NUMJ=NC*NUMD+CNOD-NUMD
     !NNNS = TOC, NWDS("OLIB" STRS CONS "NNST" 1)
     !NNNN = -LCAS*NUMD + NUMJ + NNNS
     TABLE,U(TYPE=-1):"OLIB" STRS CONS "NNST" 1 $ UPDATE DEFL CONSTR IN THIS DATASET
      OPER=XSUM: J="NNNN":"CON"
*   GO TO ENDIF
*   LABEL DERIVS $ DERIVATIVES ONLY
     !NUMI = 1
*    LABEL BDVLOOP  $ LOOP OVER DESIGN VARIABLES
*    IF("NUMI" GT "NDVA"): *GO TO EDVLOOP
      !DEL=DS,"NDIR","NOD","NUMI"(1,DUDV,CAPV,"NNST","NC")
      !CON=DEL/LIM $ DERIVATIVE OF DISPLACEMENT CONSTRAINT
      !NUMJ=NC*NUMD+CNOD-NUMD
      !NNNN = -LCAS*NUMD + NUMJ + NNNS
      TABLE,U(TYPE=-1):STRS DCON "NNST" "NUMI" $write sens to STRS DCON
       OPER=XSUM: J="NNNN": "CON"
     !NUMI = NUMI + 1
*    GO TO BDVLOOP
*    LABEL EDVLOOP
*   LABEL ENDIF
  !NC = NC + 1
*  GO TO BCASELOOP
```

```
*  LABEL ECASELOOP
* GO TO BCNODELOOP
* LABEL ECNODELOOP
*ECHO 4
$ EXITING: (DISP CONS) $ DISPLACEMENT CONSTRAINTS AND DERIVATIVES
*NOECHO 4
*RETURN
*              ENDCON
$
$-------------------------------------------------------------------------*
$
*(ADS CONS)       ADSCON
*ECHO 4
$ ENTERING: (ADS CONS)
$ ASSEMBLE CONSTRAINT AND DERIVS FOR OPTIMIZER
*NOECHO 4
$ this routine put constraints and constraint sensitivities in to tables to be
$ read by the optimizer. The ADS in names are are a remnant of when the ADS
$ optimizer was used.
$
*XQT AUS
 !NWDS = TOC, NWDS(1 G ADS 1 1)
* IF("NWDS" NE 0) : *GO TO TABLEEXISTS $ CHECK IF OPT CONSTRAINT TABLES EXIST
!SLOOP=0
*LABEL BSETLOOP
! SLOOP=SLOOP+1
* IF("SLOOP" GT "NLST") : *GO TO ESETLOOP
 !NWDX = TOC, NWDS(1 STRS CONS "SLOOP" 1)
 !NWDS = NWDS + NWDX
*GO TO BSETLOOP
*LABEL ESETLOOP
  TABLE(NI=1,NJ="NWDS",TYPE=-1) : G ADS 1 1
  TABLE(NI=1,NJ="NWDS",TYPE=-1) : "SLIB" G ADS 1 1
  TABLE(NI=1,NJ="NWDS",TYPE=-1) : "SLIB" GLIN ADS 1 1
  TABLE(NI=1,NJ="NWDS",TYPE=-1) : COG LIP 1 1
  TABLE(NI="NDVA",NJ="NWDS",TYPE=-1) : DG ADS 1 1
* LABEL TABLEEXISTS
 !NNST = 1
 !STAR = 0
* LABEL BLSETLOOP  $ loop over load sets
* IF("NNST" GT "NLST"): *GO TO ELSETLOOP
 !NWDX = TOC, NWDS(1 STRS CONS "NNST" 1)
*  IF("STATUS" EQ 2) : *GO TO DERIVS
$   CONSTRAINTS ONLY - put data from STRS CONS into G ADS
    DEFINE TEMP = "OLIB" STRS CONS "NNST" 1
    TABLE,U(TYPE=-1): G ADS 1 1
     TRANSFER(OPERATION=XSUM,SOURCE=TEMP,ILIM=1,JLIM="NWDX",DBASE="STAR")
*XQT AUS
*  IF("MINORITER" NE 0):* GO TO ENDIF
$    FIND MAXIMUM CONSTRAINT VIOLATION FOR START OF MAJOR ITER
    DEFINE TEMP = G ADS 1 1
    "ALIB" Z = MM1(TEMP)
    !CMAX = DS,1,1,1("ALIB" Z MASK MASK MASK)
*  GO TO ENDIF
*  LABEL DERIVS $ DERIVATIVES ONLY
    !DVII = 1
*    LABEL BDVLOOP $ LOOP OVER DESIGN VARIABLES
$ put sensitivities for each design variable from STRS DCON "NNST" "DVII"
```

```
$ into one big DG ADS dataset
*    IF("DVII" GT "NDVA"): *GO TO EDVLOOP
      DEFINE TEMP = STRS DCON "NNST" "DVII"
      !DVBS = STAR*NDVA + DVII-1 $ (lset#-1)*#con_1_lset*#dv + dv# -1 IS THE DBASE
      !SKIP = NDVA - 1
      TABLE,U(TYPE=-1) : DG ADS 1 1
       TRANSFER(OPER=XSUM,SOURCE=TEMP,ILIM=1,JLIM="NWDX",>
            DBASE="DVBS", DSKIP="SKIP")
    !DVII = DVII + 1
*    GO TO BDVLOOP
*    LABEL EDVLOOP
*   LABEL ENDIF
  !NNST = NNST + 1
  !STAR=STAR+NWDX
* GO TO BLSETLOOP
* LABEL ELSETLOOP
*XQT DCU
 PRINT "ALIB" Z
!CMAX
*ECHO 4
$ EXITING: (ADS CONS) $ ASSEMBLE CONSTRAINT AND DERIVS FOR ADS
*NOECHO 4
*RETURN
*              ADSCON
$
$-------------------------------------------------------------------------*
$
*(CLEA NL01)       CLEANL01
*ECHO 4
$ ENTERING: (CLEA NL01)
$ CLEAN UP LIBRARY L01
*NOECHO 4
$ this routine disables datasets no longer required
$ and then deletes them with a pack command
$
*XQT DCU
 !MARK=0
 !MARK = 1
* LABEL BSTATDELETE
   DISABLE 1 PREV DISP MASK MASK
   DISABLE 1 PREV REAC MASK MASK
   DISABLE 1 DEMP DIAG MASK MASK
   DISABLE 1 INVA KP   MASK MASK
   DISABLE 1 XINV KP   MASK MASK
   DISABLE 1 STAT DISP MASK MASK
   DISABLE 1 STAT REAC MASK MASK
   !MARK = MARK + 1
* IF("MARK" LE "NLST"): *GO TO BSTATDELETE
 DISABLE 1 IC ADS 0 0
 !MARK=0
* LABEL BAUSDELETE
   !DUMM = TOC,NWDS(1 MASK AUS  MASK MASK) MARK
*   IF("MARK" EQ -1): *GO TO EAUSDELETE
   DISABLE 1 "MARK"
* GOTO BAUSDELETE
* LABEL EAUSDELETE
 PACK 1
*ECHO 4
```

```
$ EXITING: (CLEA NL01) $ CLEAN UP L01 LIBRARY
*NOECHO 4
*RETURN
*              CLEANL01
$-------------------------------------------------------------------------*
$
*(CALC MASS)      ENDMASS
*ECHO 4
$ENTERING: (CALC MASS)
*NOECHO 4
$ routine which controls mass calculation of finite element model
$ calls MASS GRP1 for 1D elems and MASS GRP2 for 2D elems
!MASS=0.
* IF("GE21" EQ 0): *GO TO 8900
  !NAM5='E21
  !IE=6
*  DCALL(MASS GRP1) $ CALCULATE E21 MASS
* LABEL 8900
  !IE=2
* IF("GE23" EQ 0): *GO TO 8901
  !NAM5='E23
*  DCALL(MASS GRP1) $ CALCULATE E23 MASS
* LABEL 8901
* IF("GE31" EQ 0): *GO TO 8902
  !NAM5='E31
*  DCALL(MASS GRP2) $ CALCULATE E31 MASS
* LABEL 8902
* IF("GE32" EQ 0): *GO TO 8903
  !NAM5='E32
*  DCALL(MASS GRP2) $ CALCULATE E32 MASS
* LABEL 8903
* IF("GE33" EQ 0): *GO TO 8904
  !NAM5='E33
*  DCALL(MASS GRP2) $ CALCULATE E33 MASS
* LABEL 8904
* IF("GE41" EQ 0): *GO TO 8905
  !NAM5='E41
*  DCALL(MASS GRP2) $ CALCULATE E41 MASS
* LABEL 8905
* IF("GE42" EQ 0): *GO TO 8906
  !NAM5='E42
*  DCALL(MASS GRP2) $ CALCULATE E42 MASS
* LABEL 8906
* IF("GE43" EQ 0): *GO TO 8907
  !NAM5='E43
*  DCALL(MASS GRP2) $ CALCULATE E43 MASS
* LABEL 8907
* IF("GE44" EQ 0): *GO TO 8908
  !NAM5='E44
*  DCALL(MASS GRP1) $ CALCULATE E44 MASS
* LABEL 8908
!IE=FREE()
!NAM5=FREE()
*ECHO 4
$ EXITING: (CALC MASS)
*NOECHO 4
*RETURN
*              ENDMASS
```

```
$
$-------------------------------------------------
$
*(MASS GRP1)      ENDMGRP1
*ECHO 4
$ENTERING: (MASS GRP1)
*NOECHO 4
$ routine calculated mass of various 1D element types by extracting
$ element lengths, areas, and densities from element efile
*XQT EI1
 EXTRACT
  SOURCE="NAM5"
  CONTENT SPEC
   GEOM 1 : CREATE "ALIB" GEOM ELEM $table of elem lengths
  CONTENT SPEC
   SECT "IE" : CREATE "ALIB" SECT ELEM $table of elem areas
  CONTENT SPEC
   MATE 4 : CREATE "ALIB" DENS ELEM $table of elem densities
*XQT AUS
  INLIB="ALIB" : OUTLIB="ALIB"
  DEFI X=SECT ELEM :DEFI Y=DENS ELEM :DEFI Z=GEOM ELEM
  UWT ELEM = PROD(X,Y)
  GRP WT = XTY(Z,UWT)
!GWT=DS,1,1,1("ALIB",GRP ,WT,MASK,MASK)
!MASS = MASS + GWT
!GWT=FREE()
*ECHO 4
$EXITING: (MASS GRP1)
*NOECHO 4
*RETURN
*              ENDMGRP1
$-------------------------------------------------
$
*(MASS GRP2)      ENDMGRP2
*ECHO 4
$ENTERING: (MASS GRP2)
*NOECHO 4
$ routine calculated mass of various 2D element types by extracting
$ element areas and unit weights from element efile
*XQT EI1
 EXTRACT
  SOURCE="NAM5"
  CONTENT SPEC
   GEOM 1 : CREATE "ALIB" GEOM ELEM $table of element areas
  CONTENT SPEC
   SECT 2 : CREATE "ALIB" SECT ELEM $table of element unit weights
*XQT AUS
  INLIB="ALIB" : OUTLIB="ALIB"
  DEFI X=SECT ELEM : DEFI Z=GEOM ELEM
  GRP WT = XTY(Z,X)
!GWT=DS,1,1,1("ALIB",GRP ,WT,MASK,MASK)
!MASS = MASS + GWT
!GWT=FREE()
*ECHO 4
$EXITING: (MASS GRP2)
*NOECHO 4
*RETURN
*              ENDMGRP2
```

```
$-----------------------------------------------------
$
*(ELEM MBED)       ENDEMBED
*ECHO 4
$ ENTERING (ELEM MBED)
*NOECHO 4
$ routine which takes datasets produced by processor MBED
$ tables of elements in group
$ tables of element section properties
$ and embeds the section property of perturbed elems into EFILE
$
$  LOOP THRU ITEMS IN EMBE GRP DATASET
 !NENT = TOC,NJ("ALIB"  EMBE GRP  MASK MASK)
!NGRP = 0
*LABEL GRPLOOP
!NGRP = NGRP +1
*IF("NGRP" GT "NENT"): *GO TO EGRLOOP
!EXX=DS,1,"NGRP",1("ALIB",EMBE,GRP,MASK,MASK) $ GET ELEMENT TYPE
!GRP=DS,2,"NGRP",1("ALIB",EMBE,GRP,MASK,MASK) $ GET GROUP NUMBER
!END = TOC,NI("ALIB" EMBE EFIL 1 1)
!END = END - 5
*XQT EI1
  EMBED
  SOURCE = "ALIB" EMBE EFIL "NGRP"
  CONTENT SPECIFICATION
   MATERIAL, 1, 6
   SECTION, 2, "END"
  DEST = "EXX" "GRP"
* GOTO GRPLOOP
*LABEL EGRLOOP
*XQT DCU
*ECHO 4
$ EXITING: (ELEM MBED)
*XQT DCU
*NOECHO 4
*RETURN
*              ENDEMBED
$
$-------------------------------------------------------------------------*
$
*(CALC PEN)    ENDPEN
*ECHO 4
$ ENTERING: (CALC PEN)
$ COMPUTE PENALTY FOR THE VIOLATED CONSTRAINTS
*NOECHO 4
$
*XQT AUS
 INLIB ="ALIB"
 OUTLIB ="ALIB"
$
$  find number of violated constraints
  !LENG = TOC,NWDS(1 G ADS 1 1)
  TABLE(NI=1,NJ="LENG"): DUMM : INIT= 0.0 $ A 0% CUTOFF
  DEFINE G = 1 G ADS 1 1  $the constraint dataset
  JZER = JLIST(ANY G GT DUMM)
  !LENZ = TOC,NWDS("ALIB" JZER MASK MASK MASK) $ # of violations
  !GLARGE = 0.0
*IF("LENZ" EQ 0):*GOTO NOPEN  $ if no constraints violated - no penalty
```

```
$
$ compute the penalty
 DE1
  SOURCE = 1 G ADS 1 1 : DEST = G ADS 1 1
  JS = JZER : EX1
 DEFINE G = G ADS 1 1
 TABLE(NI=1,NJ=1,TYPE=-1): TMP2 $ SUM OF THE VIOLATED CONSTRAINTS
  TRANSFER(SOURCE=G, JLIM="LENZ",DSKIP =-1,OPER = SUM)
 EXIT
 !GLARGE = DS,1,1,1("ALIB" TMP2 AUS MASK MASK) $sum of violated constraints
 !GLARGE = GLARGE*KPEN $ HERE ASSUME THAT LARGEST LM < KPEN
$
*LABEL NOPEN
!NEWPEN = DS,1,1,1(1 F ADS 1 1)  $ objective (total mass)
!NEWPEN = NEWPEN + GLARGE       $ objective + penalty
$
*ECHO 4
$ EXITING: (CALC PEN) $ COMPUTE PENALTY FOR VIOLATED CONSTRAINTS
*NOECHO 4
$
*RETURN
*              ENDPEN
$
$-------------------------------------------------------------------------*
$
*(UPDA GRAD)  ENDUPDA
*ECHO 4
$ ENTERING: (UPDA GRAD)
$ UPDATE CONSTRAINT GRADIENTS USING APPROX. METHOD
*NOECHO 4
$
* XQT AUS
 ERASE "ALIB"
 INLIB="ALIB"
 OUTLIB="ALIB"
 DEFINE G = 1 G ADS 1 1  $ Constraints form CONS with new DVs
 DEFINE GLIN = "SLIB" GLIN ADS $ Linear approx from optimizer
 DEFINE X = 1 X ADS 1 1  $ New DVs from optimizer
 DEFINE XOLD = "SLIB" X ADS 1 1 $ DVs from prior to optimization
 DELG = SUM(G, -1. GLIN) $ Difference of calculated and linear cons
 DX = SUM(X, -1. XOLD)  $ Change in DVs during optimization
 DEFINE DG = 1 DG ADS 1 1
 DGDG = UNION(DG)
 TABLE,U(TYPE=-1): DGDG   $ create table of DG**2 terms
  TRANSFER(SOURCE=DG,OPER=MULTIPLY)
 DXDX = UNION(DX)
 TABLE,U(TYPE=-1): DXDX   $ create table of DX**2 terms
  TRANSFER(SOURCE=DX,OPER=MULTIPLY)
 DENM=RPROD(DXDX,DGDG) $ 1 BY NCON  vector of DX**2 * DG**2
$ compute updates to constraint derivatives
 !NCON = TOC,NJ("ALIB" DENM MASK MASK MASK)       $
 TABLE(NI=1,NJ = "NCON"): DUM : INITIAL = 1.E-14   $ AVOID DIVIDE BY
 TABLE,U(TYPE=-1): DENM                    $ ZERO
  TRANSFER(SOURCE=DUM,OPER=SUM)                $
 TABLE,U(TYPE=-1): DELG
  TRANSFER(SOURCE = DENM,OPER=DIVIDE)
 DXT = RTRAN(2. DX) $  NDVI BY 1
 AIJ = RPROD(DXT,DELG)
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 1996	Contractor Report

4. TITLE AND SUBTITLE
Documentation for a Structural Optimization Procedure Developed Using the Engineering Analysis Language (EAL)

5. FUNDING NUMBERS
C NAS1-19000

WU 537-06-34-20

6. AUTHOR(S)
C. J. Martin, Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Lockheed Martin Engineering and Sciences
144 Research Drive
Hampton, VA 23666

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23682-0001

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
NASA CR-201632

11. SUPPLEMENTARY NOTES
Langley Technical Monitor - Stephen J. Scotti

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category 39

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

This report describes a structural optimization procedure developed for use with the Engineering Analysis Language (EAL) finite element analysis system. The procedure is written primarily in the EAL command language. Three external processors which are written in FORTRAN generate equivalent stiffnesses and evaluate stress and local buckling constraints for the sections. Several built-up structural sections were coded into the design procedures. These structural sections were selected for use in aircraft design, but are suitable for other applications. Sensitivity calculations use the semi-analytic method, and an extensive effort has been made to increase the execution speed and reduce the storage requirements. There is also an approximate sensitivity update method included which can significantly reduce computational time. The optimization is performed by an implementation of the MINOS V5.4 linear programming routine in a sequential liner programming procedure.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Structural Optimization, Structural Sensitivity Analysis	63
	16. PRICE CODE: A04

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	

```
  TABLE,U(TYPE=-1): AIJ
   TRANSFER(SOURCE=DGDG,OPER=MULTIPLY)
$ add updates to constraint data
  TABLE,U(TYPE=-1):1 DG ADS 1 1
   TRANSFER(SOURCE=AIJ,OPER=SUM)
  EXIT
!EXACT = 0
$
*ECHO 4
$ EXITING: (UPDA GRAD) $ UPDATE CONSTRAINT GRADIENTS USING APPROX. METHOD
*NOECHO 4
$
*RETURN
*              ENDUPDA
$-------------------------------------------------------------------------*
$
*(SENS FILT)   ENDFILT
*ECHO  4
$ ENTERING: (SENS FILT)
$ CHOOSE WHICH ELEMENT GROUPS NOT TO CALCULATE STRESS SENSITIVITIES FOR
*NOECHO 4
$
*XQT AUS
 INLIB= "ALIB"
 OUTLIB= "ALIB"
!NACT=TOC,NJ (1 COMP GRP MASK MASK)
!LOOP=0
*LABEL GLOOP   $ LOOP THRU COMP GROUP DATASET
  !LOOP=LOOP+1
*IF("LOOP" GT "NACT"): *GO TO EGLOOP
  !EXX=DS,3,"LOOP",1 (1,COMP GRP 1 1)
  !NGRP=DS,4,"LOOP",1 (1,COMP GRP 1 1)
$  find maxixum constraint value
   DEFINE G = 1 "EXX" CON "NNST" "NGRP"  $ element constraint dataset
   JMAX=MM1(G)
   !MAX = DS,1,1,1,("ALIB" JMAX MASK MASK MASK) $ # of violations
*IF("MAX" GT "GLIM"):*GOTO NOCHNG  $ constraints > glim so calc stresses
  TABLE,U(TYPE=0):1 GACT "EXX"
    I=1: J="NGRP": 0   $ else don't calc stresses
*LABEL NOCHNG
*GO TO GLOOP
*LABEL EGLOOP
*ECHO 4
$ EXITING: (SENS FILT) $ CHOOSE ELEMENT GROUPS NOT TO CALCULATE STRESS SENS FOR
$
*RETURN
*              ENDFILT
$
$-------------------------------------------------------------------------*
*DCALL(DRIV OPT)
*XQT DCU
 COPY 1 3 DR X 1 1
 COPY 1 3 DR DIF 1 1
*XQT EXIT
```